



OPEN

# Binding affinity predictions with hybrid quantum-classical convolutional neural networks

L. Domingo<sup>1,2,3,4</sup>✉, M. Djukic<sup>4</sup>, C. Johnson<sup>4</sup> & F. Borondo<sup>3</sup>

Central in drug design is the identification of biomolecules that uniquely and robustly bind to a target protein, while minimizing their interactions with others. Accordingly, precise binding affinity prediction, enabling the accurate selection of suitable candidates from an extensive pool of potential compounds, can greatly reduce the expenses associated to practical experimental protocols. In this respect, recent advances revealed that deep learning methods show superior performance compared to other traditional computational methods, especially with the advent of large datasets. These methods, however, are complex and very time-intensive, thus representing an important clear bottleneck for their development and practical application. In this context, the emerging realm of quantum machine learning holds promise for enhancing numerous classical machine learning algorithms. In this work, we take one step forward and present a hybrid quantum-classical convolutional neural network, which is able to reduce by 20% the complexity of the classical counterpart while still maintaining optimal performance in the predictions. Additionally, this results in a significant cost and time savings of up to 40% in the training stage, which means a substantial speed-up of the drug design process.

The ability to predict the binding affinity between a potential drug and its target protein is important at a fundamental level<sup>1</sup>, and also crucial for the success of drug discovery at the early stages of drug design<sup>2</sup>. The theoretical computation of the binding affinities still needs further development<sup>3</sup>, and the experimental determination for a large number of small molecules and their targets is time-consuming and expensive<sup>4</sup>. As a result, alternative machine learning methods that can make accurate predictions have been greatly welcomed, and then widely used in this field.

Traditional methods are physics-based, meaning that they rely on biophysical models of the proteins-ligand structure to estimate their binding affinity. Various strategies exist within the realm of physics-based methods. For instance, all-atom molecular dynamics methods<sup>1,4-6</sup> simulate the temporal behavior of drug-protein complexes to estimate binding affinities. Unfortunately, such rigorous methods are computationally expensive and often require a lot of expert knowledge and domain expertise<sup>7,8</sup>. Quantum mechanical calculations, encompassing semiempirical, density-functional theory, and coupled-cluster approaches, have also been employed for binding affinity predictions<sup>1</sup>. Although these methods tend to offer high accuracy, their applicability is hindered by the size and complexity of protein-ligand structures, making it impractical to study larger molecules. Finally, force-field scoring functions are also used to evaluate the energy associated with the complex formed by the ligand and protein<sup>9-11</sup>. The scoring function considers bonded and non-bonded interactions between the ligand and protein, including electrostatic interactions, van der Waals interactions, and bonding terms. While these methods are less time-consuming compared to previous approaches, they sacrifice some accuracy in their predictions.

Machine learning techniques, and more specifically deep learning methods, have recently attracted attention for their ability to improve upon traditional physics-based methods. Traditional methods need to be adapted to every single protein-ligand pair, requiring long computations and domain expertise for every single binding affinity prediction. With the recent available big datasets, making individual predictions is no longer worth it. On the other hand, data-based methods such as machine learning require only a one-time training process, enabling them to provide predictions indefinitely with minimal computational cost. Therefore, the development of efficient and accurate data-based methods is crucial for the development of drug design.

<sup>1</sup>Grupo de Sistemas Complejos, Universidad Politécnica de Madrid, 28035 Madrid, Spain. <sup>2</sup>Instituto de Ciencias Matemáticas (ICMAT), Campus de Cantoblanco UAM, Nicolás Cabrera, 13-15, 28049 Madrid, Spain. <sup>3</sup>Departamento de Química, Universidad Autónoma de Madrid, 28049 Cantoblanco, Madrid, Spain. <sup>4</sup>Ingenii Inc., New York, USA. ✉email: laia@ingenii.dev

Unlike other traditional machine learning methods, deep learning can learn directly from the atomic structure of the protein–ligand pair without relying on hand-curated, manually-extracted features from the data. A commonly-used deep learning approach for binding affinity prediction is the three-dimensional convolutional neural network (3D CNN)<sup>12–17</sup>. These networks represent atoms and their properties in a 3D space taking into account the local 3D molecular structure and the relationships between atoms. The 3D representations used as input of the 3D CNN are high-dimensional matrices since millions of parameters are required to describe only one data sample. Because of this high dimensionality, a complex deep learning model is required to uncover all the hidden patterns that can help predict the binding affinity. Training such a model means finding the optimal value for parameters that minimize a suitable loss function. More complex models, having more training parameters, require longer execution times, which limits the exploration of different architectures or hyperparameters. This training process can be heavily accelerated using powerful GPUs. However, as the size of datasets continues to grow, it is crucial to not only scale computational resources, like GPUs but also enhance the efficiency of our algorithms to meet the demands of larger datasets. Therefore, there is a pressing need to discover more efficient training approaches for these networks, which would enable the exploration of novel network architectures and innovative pre-processing techniques. This advancement holds the promise of developing highly accurate models that not only accelerate the drug design process but also instil greater confidence in the reliability of computational methods.

The complexity of a machine learning model also affects its generalisation capacity. According to Hoeffding's theorem<sup>18</sup>, highly complex machine learning models require a large amount of data to reduce the variance of the model predictions, as stated by Hoeffding's inequality

$$E_{\text{out}} \leq E_{\text{in}} + \mathcal{O}\left(\sqrt{\frac{K}{N_{\text{samples}}}}\right), \quad (1)$$

where  $E_{\text{out}}$  is the error in the test set,  $E_{\text{in}}$  is the error in the training set,  $K$  is a measure of the complexity of the model and  $N_{\text{samples}}$  is the number of data samples, which should be at least comparable to the complexity of the model to guarantee low errors in the predictions of new data. In some cases, when the test data is similar enough to the training data, a smaller training set can still allow for good performance. Nonetheless, increasing the complexity of a model always increases its chances of producing overfitting, and thus it is convenient to resort to simpler machine learning models.

Quantum machine learning methods have the potential to solve numerical problems exponentially faster than classical methods<sup>19</sup>. Although fault-tolerant quantum computers are still not available, a new trend of quantum algorithms, called noisy intermediate-scale quantum (NISQ)<sup>20</sup> era is devoted to designing quantum algorithms that provide a quantum advantage with the quantum computers available today. Because of the exponential growth scaling of the Hilbert space dimension, quantum computers can process large amounts of data with few qubits<sup>21</sup>. It is for this reason that combining quantum algorithms with machine learning allows for reducing the complexity of the classical machine learning methods while maintaining their accuracy. Managing large datasets is crucial in the field of drug design, considering the remarkable growth in available data for computational studies. As an illustration, the PDBbind dataset, which will be utilized in this study (refer to “Data” section), included a mere 800 complexes in the general set back in 2002. However, by 2016, this number had risen to approximately 9000 samples. The current version (2020) comprises over 14,000 samples, and it is anticipated to grow by 20% this year<sup>22</sup>. Consequently, it is imperative to develop efficient machine learning models capable of handling such vast datasets to effectively address the data revolution. Hybrid quantum-classical machine learning models have emerged as a promising approach that may hold the key to designing efficient models capable of effectively working with these large amounts of data.

In this paper, we propose a hybrid quantum-classical 3D CNN, which by replacing the first convolutional layer with a quantum circuit, effectively reduces the number of training parameters of the model. Our results show that as long as the quantum circuit is properly designed, the hybrid CNN maintains the performance of the corresponding classical CNN. Moreover, the hybrid CNN requires 20% fewer training parameters and the training times are reduced by 20–40%, depending on the hardware used for training. We extensively investigate various architectures for the quantum layer of the hybrid CNN, revealing that the design of this quantum layer plays a crucial role in achieving optimal performance. Therefore, this study offers an optimal strategy for designing hybrid CNNs that excel in machine learning tasks, providing valuable insights for future advancements in this field.

It is worth noting that all the quantum circuits used in this work have been executed using quantum simulation due to the current limitations of quantum hardware. The quantum layer has been modified and optimized to run effectively on a GPU, just as the classical 3D CNN. Specifically, the quantum circuits within the quantum layer correspond to quantum unitary transformations, that were constructed using PyTorch<sup>23</sup> tensors. This design choice enables them to be integrated into the neural network's computational graph, facilitating efficient optimization on a GPU. Even though the training was done with classical resources, we additionally provide performance benchmarks considering different noise models and error probabilities in our models. Our results show that with error probabilities lower than  $p = 0.01$  and circuits with 300 gates, a common error mitigation algorithm, namely *data regression error mitigation*<sup>24</sup>, can accurately mitigate the errors produced by the quantum hardware.

Our findings clearly indicate that hybrid quantum-classical machine learning methods have the potential to speed up the training process of classical machine learning methods and reduce the computational resources needed to train them.

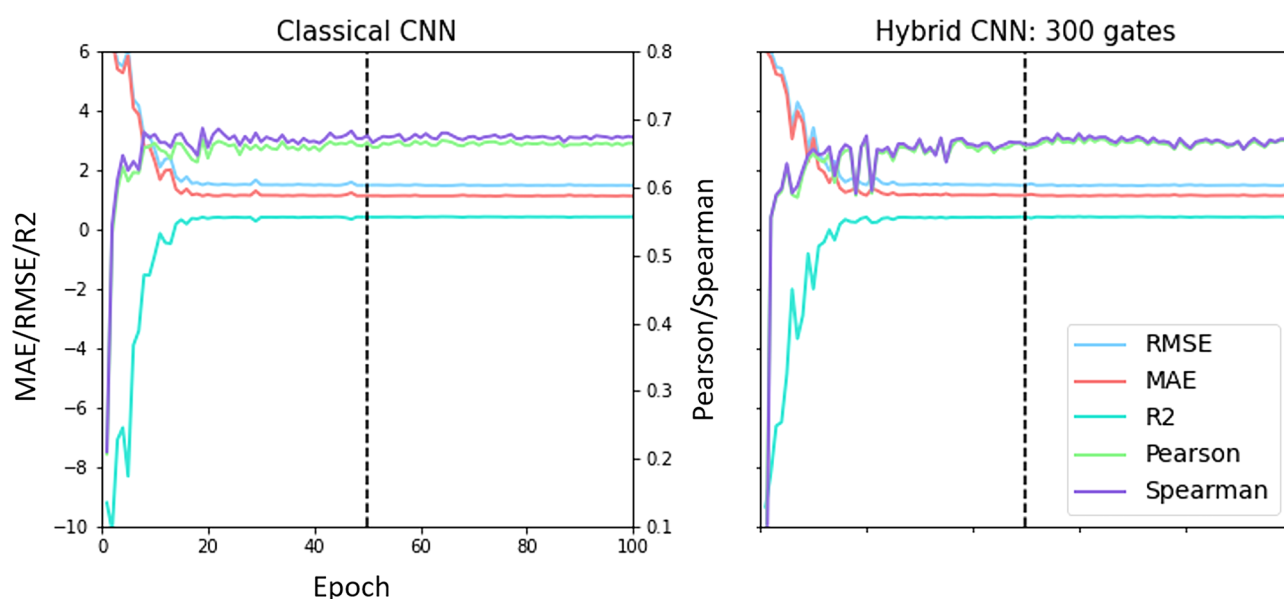
## Results

In this section, we present the results obtained with the classical CNN and the different variations of the hybrid CNN (see “Classical CNN” and “Hybrid quantum-classical CNN” sections for more details). The performance of the models is evaluated against the core set of the 2020 PDBBind dataset (see “Data” section). The training and validation steps are done with the refined set (separated into training and validation sets) for all the models. To reduce overfitting of the training data, we use an early stopping procedure, finishing the training step when the performance in the validation set has converged. To evaluate the convergence of the training process, five error metrics are considered:

- *Root mean squared error (RMSE)*
- *Mean absolute error (MAE)*
- *Coefficient of determination R squared (R2)*: proportion of the variation of the dependent variable (binding affinity) that is predictable from the independent variable (prediction of the model).
- *Pearson correlation coefficient (Pearson)*: Linear correlation between two variables (binding affinity and prediction of the model). It ranges between  $-1$  and  $+1$ .
- *Spearman coefficient*: Monotonic correlation coefficient. It ranges between  $-1$  and  $+1$ . A Spearman correlation of  $+1$  or  $-1$  occurs when a variable is a perfect monotone function of the other.

Figure 1 shows the evolution of the five error metrics in the validation set, for the classical CNN and the hybrid CNN with 300 quantum gates. We see that for both cases, all the error metrics stabilize after 50 epochs. Further training the models with the same data could lead to overfitting of the training data, thus decreasing the generalization capacity. For this reason, the training of all the models was stopped at 50 epochs. The results in Fig. 1 also show that the Pearson and Spearman coefficients oscillate more than the other error metrics, even when the training has converged. For this reason, we conclude that, in this case, the RMSE, MAE and R2 are better measures of model convergence.

Once the models have been trained, we evaluate their performance on the test set. Figure 2 shows the results obtained for the five error metrics, evaluated on the test set, for all models studied in this work. We compare the performance of the hybrid models with 20–600 quantum gates (histogram), with the performance of the classical CNN (horizontal dashed orange line). The results show that, in general, the performance of the hybrid CNN models increases with the number of quantum gates until roughly the same performance as the classical CNN is reached, at 300 quantum gates. From that point, the models with 400, 500 and 600 gates oscillate around the classical performance and do not significantly improve with the number of quantum gates. Therefore, we conclude that the number of quantum gates does affect the performance of the model, for shallow quantum circuits, and that it stabilizes when the quantum circuits achieve a certain depth. The minimal number of gates needed to achieve classical performance, in this case, is around 300 quantum gates. Thus, for a certain choice of quantum circuits, decreasing the complexity of the CNN does not decrease its predictive performance. To further validate this claim, Fig. 2 contains the performance of a downsized classical CNN. In this modified model, the initial convolutional layer has been entirely omitted. Despite having an equivalent number of training parameters as the hybrid CNN models, the performance on the core set, as evaluated using the five error metrics under consideration, falls notably short of that achieved by the standard classical CNN and the most effective hybrid CNN



**Figure 1.** Error metrics evaluated in the validation set as a function of the training epochs for the classical CNN (left) and the hybrid CNN with 300 gates (right). Both models are seen to converge after 50 epochs.



**Figure 2.** Evaluation of the five error metrics in the core set and comparison of the hybrid CNN models constructed from the Ising model and the G3 family with 20,50,100,200,300,400,500 and 600 gates, together with the classical CNN results (horizontal dashed orange line). Additionally, the performance of a smaller classical CNN model, with the same number of training parameters as the hybrid CNN, is given (horizontal dotted purple line). All models have been trained with the refined set.

models. Hence, merely reducing the size of the classical CNN model does not suffice to preserve its predictive capabilities. Achieving optimal performance while reducing training time necessitates the incorporation of an

additional quantum layer. Figure 2 also shows the performance of the hybrid model constructed from the Ising model, which is a bit worse than the one for our optimal G3 hybrid model.

The main motivation for designing a hybrid CNN model was to reduce the complexity and thus the training time of the neural network. A measure of complexity that does not depend on the hardware where the model is trained is the number of training parameters. Table 1 shows the training parameters of the classical CNN and all the hybrid CNN models used in this work. As can be seen, the classical CNN uses around 10 million parameters, while the hybrid CNNs only use around 8 million parameters, demonstrating a 20% reduction in model complexity. Notice that the number of quantum gates of the quantum layer does not affect the number of training parameters of the network, since the parameters of the quantum circuit are carefully selected and fixed during training. This is precisely one of the advantages of using quantum reservoirs as the quantum transformation. On the other hand, the training times depend on the hardware where the training is executed. Table 1 shows that training the CNNs with only CPUs requires much longer execution times. Using GPUs highly accelerates the training process, thus reducing the training time from many days to hours. In our experiments, the models have been trained using only CPUs and with two types of GPUs. Details of the used hardware are shown in Table 2.

As can be seen in Table 1, the improvement in training times of the hybrid model over the classical varies from 26 to 42%. Using more powerful GPUs reduces the difference in training times, but the hardware is also more expensive. The difference in training times in all cases is limited by the difference in training parameters, which is a hardware-agnostic measure of complexity.

After analyzing the results from the models trained with the refined set, we repeated the experiments training the models with the general set. The general set has almost three times more data than the refined set, and thus the training takes more time and computational resources. We observed that the models required more epochs for the performance to converge in the validation set. The performance metrics evaluated in the test set are displayed in Fig. 3. We see that the performance results are equivalent to the ones from the models trained with the refined set. The performance of the hybrid G3 models increases with the number of quantum gates until it converges at around 300 gates. Then, the performance oscillates around the classical performance. The Ising model has suboptimal performance compared to the classical CNN or the hybrid G3 CNN with 300 gates. From these results, we conclude that training the models with the general set leads to equivalent results to training the models with the refined set, but it requires longer training times and more computational resources.

CNNs are widely used models to learn from data such as time series, images or volumetric representations. Their goal is to unravel hidden patterns from the input data and use them to predict the target. Thus, the complexity of a CNN model highly depends on the complexity of the data. Hybrid quantum-classical CNN models can help reduce the number of parameters of the neural network while maintaining its prediction capacity. One natural question that arises here, is how this reduction of training parameters scales with the size of the data. Let us consider that each sample has size  $(C, N, N, N)$ , where  $C$  is the number of features and  $N$  is the size of the volume side. The reduction of model complexity corresponds to the number of parameters of the first layer of the network. Therefore, the reduction of training parameters scales linearly with the number of features  $C$ . The number of training parameters does not explicitly depend on  $N$ , since each filter is applied locally to a portion of the data, as many times as needed to cover the whole sample. However, when the dimensionality of the data increases, usually more filters are needed for the CNN to converge. As the data complexity increases, more complex models are needed to learn useful information from it.

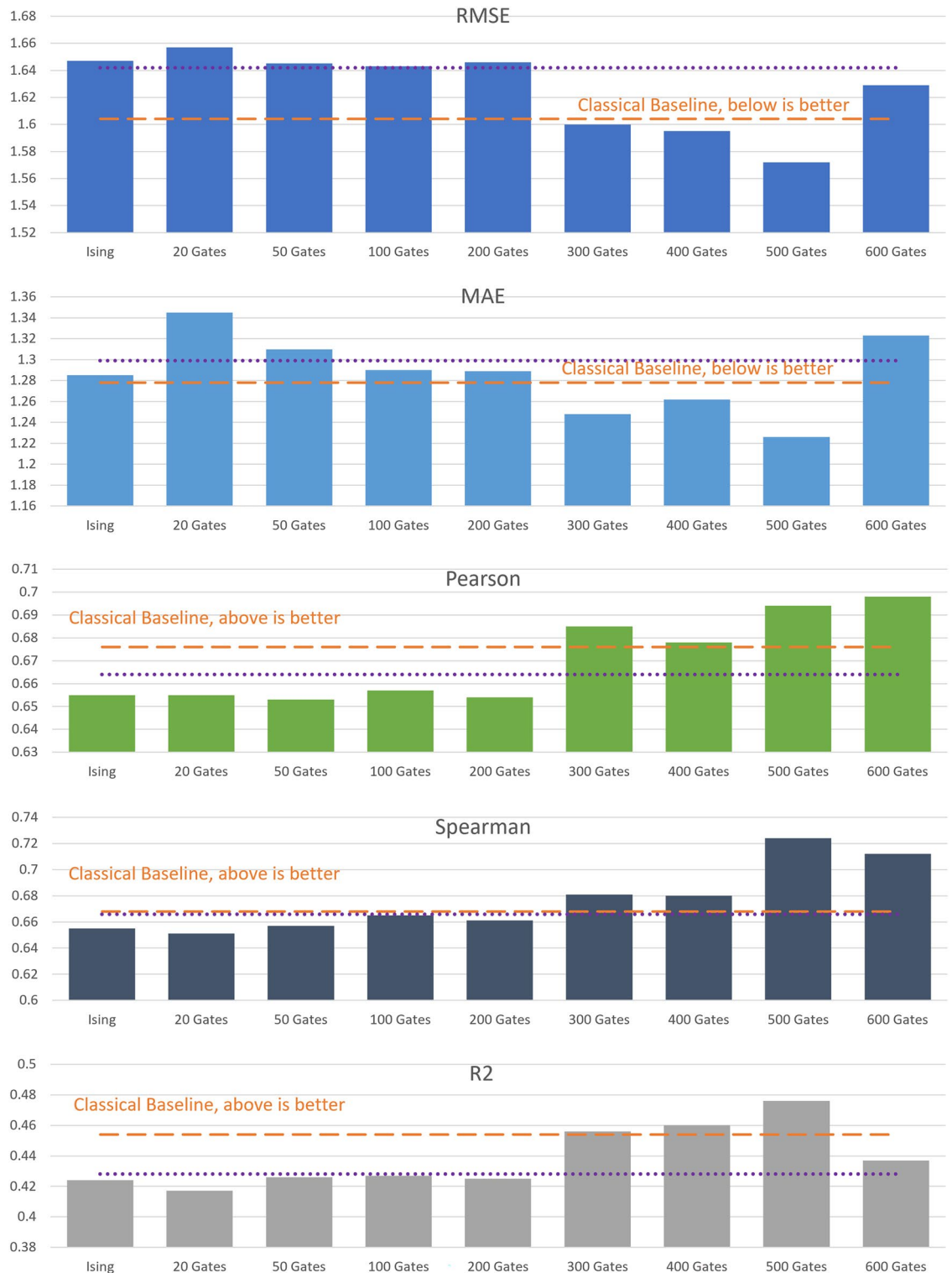
After analyzing the results of the noiseless quantum circuits, we perform noisy simulations for three different quantum channels and analyze the corresponding performance. An example of the output of a quantum circuit for different noise models and different error probabilities is shown in Fig. 4. We see that the three noise models reduce the probability amplitude of the circuits' outputs. The main difference between the behavior of the noise models is that the phase damping channel reduces the probability amplitude slower than the other two models.

Hardware	Hybrid CNN	Classical CNN	Difference (%)
Azure CPU	10.7 days	18.3 days	42
Azure GPU	24 h	39 h	38
Purdue Anvil GPU	16.3 h	22.1 h	26
Number of training parameters	8,088,499	10,137,129	20

**Table 1.** Training time and number of training parameters for the classical and hybrid 3D CNN trained with the refined set of the PDBBind dataset.

	Azure CPU	Azure GPU	Purdue Anvil GPU
CPU	Intel Xeon E5-2673 v3 2.4 GHz	Intel Xeon E5-2690 v3	2 × 3rd Gen AMD EPYC 7763
Cores	4	6	128
GPU	–	1 × NVIDIA Tesla K80	1 × NVIDIA A100
Memory size	14 GB	56 GB	512 GB

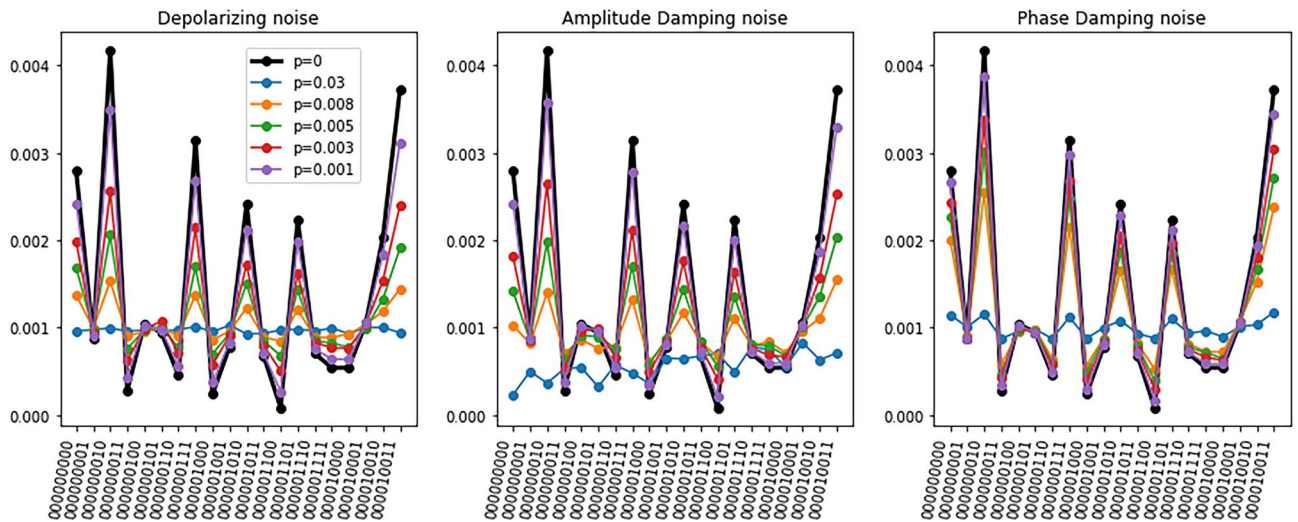
**Table 2.** Hardware specification for the different devices used to train the classical and hybrid 3D CNNs.



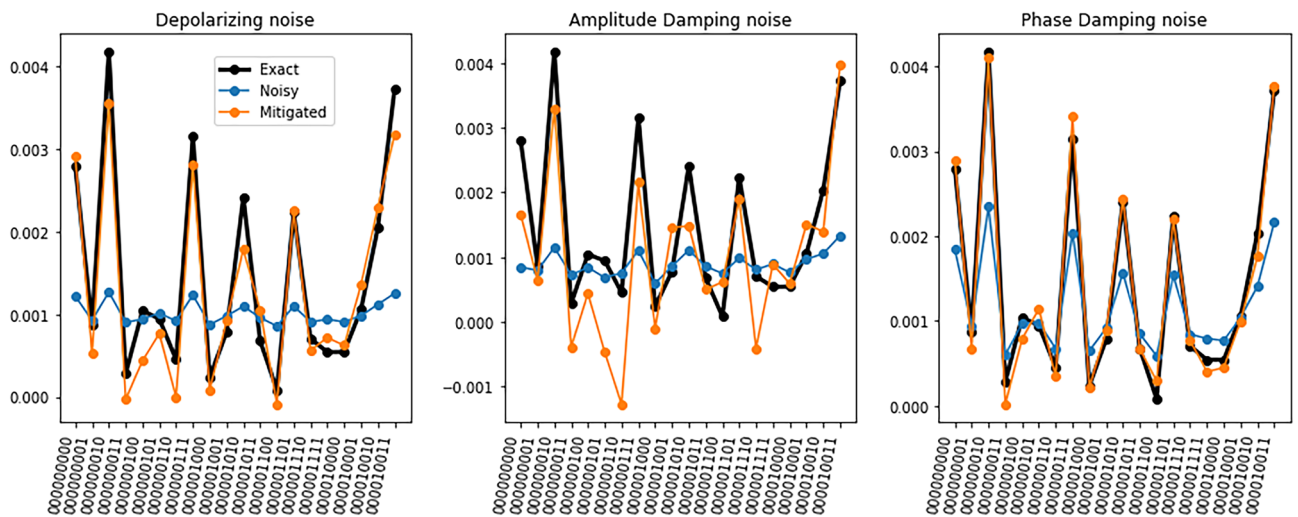
**Figure 3.** Same as Fig. 2, trained with the general set.

In all cases, when the error probability reaches  $p = 0.03$ , the quantum information is lost, since the amplitude peaks can no longer be distinguished. On the other hand, when the error rate is smaller than  $p = 0.03$ , the DRER algorithm can successfully mitigate the noisy outputs. An example of the performance of the DRER algorithm for  $p = 0.01$  is shown in Fig. 5. Even though the noise of the quantum device significantly reduces the amplitudes of the distribution, the DRER algorithm can recover the original amplitudes with significant accuracy.

For every noise model and error rate  $p$ , we performed a hyperparameter optimization to obtain the best linear model to mitigate the quantum errors. The results are shown in Table 3. In addition to evaluating the



**Figure 4.** Example of an output of the quantum circuit used in the quantum convolutional layer for three noise models and different error rates. For easier visualization, only the first 20 outputs are displayed in the figure.



**Figure 5.** Example of an output of the quantum circuit used in the quantum convolutional layer for three noise models and error rate  $p = 0.01$ , together with the output of the error mitigation algorithm. For easier visualization, only the first 20 outputs are displayed in the figure.

mean squared error (MSE), we also evaluate the tendency accuracy, that is, the fraction of times the DRER algorithm modifies the output in the correct direction. Let  $y_{noisy}$ ,  $y_{noiseless}$ ,  $y_{mitigated}$  be the noisy, noiseless and mitigated counts respectively. Then, the tendency accuracy measures the proportion of times  $|y_{mitigated} - y_{noiseless}| < |y_{noisy} - y_{noiseless}|$ .

Table 3 shows that when  $p \leq 0.01$  the MSE of the mitigated circuits is smaller than the MSE of the noisy circuits. On the other hand, for  $p = 0.03$  the MSE of the mitigated circuits is similar or even larger than the MSE of the noisy circuits, and the tendency accuracy is barely better than random guessing. This result agrees with the results in Fig. 4 since the noisy simulations with  $p = 0.03$  are basically a constant value. Table 3 also shows that the tendency accuracy increases as the error probability  $p$  decreases.

For the depolarizing quantum channel, the tendency accuracy reaches the value 0.8 when  $p = 0.008$  and increases to 0.89 with  $p = 0.001$ . The amplitude damping noise seems to be the hardest to mitigate since the tendency accuracy increases slower than the tendency accuracy of the other noise models. This is because the amplitude damping channel introduces non-zero counts apart from mitigating the amplitudes of the noiseless simulation<sup>25</sup>. On the other hand, the tendency accuracy increases faster with the phase damping channel, when it reaches the value 0.81 with  $p = 0.01$ . All in all, these results show that as long as the error rates are smaller than  $p = 0.01$ , the DRER algorithm can successfully mitigate the errors introduced by the quantum device on the quantum convolutional layer with 300 gates.

Error model	Error rate $p$	Regularization $\alpha$	MSE noisy circuits	MSE mitigated circuits	Tendency accuracy
Depolarizing	0.030	$10^{-1}$	$9.7 \cdot 10^{-7}$	$9.7 \cdot 10^{-7}$	0.59
	0.010	$10^{-2}$	$7.7 \cdot 10^{-7}$	$2.1 \cdot 10^{-7}$	0.75
	0.008	$10^{-5}$	$6.7 \cdot 10^{-7}$	$1.0 \cdot 10^{-7}$	0.80
	0.005	$10^{-5}$	$4.3 \cdot 10^{-7}$	$3.4 \cdot 10^{-8}$	0.84
	0.003	$10^{-5}$	$2.3 \cdot 10^{-7}$	$1.1 \cdot 10^{-8}$	0.87
	0.001	$10^{-6}$	$4.0 \cdot 10^{-8}$	$1.4 \cdot 10^{-9}$	0.89
Amplitude damping	0.030	$10^{-1}$	$1.1 \cdot 10^{-6}$	$9.9 \cdot 10^{-7}$	0.54
	0.010	$10^{-4}$	$7.3 \cdot 10^{-7}$	$5.3 \cdot 10^{-7}$	0.62
	0.008	$10^{-4}$	$6.2 \cdot 10^{-7}$	$3.4 \cdot 10^{-7}$	0.66
	0.005	$10^{-5}$	$3.9 \cdot 10^{-7}$	$6.8 \cdot 10^{-8}$	0.75
	0.003	$10^{-5}$	$2.0 \cdot 10^{-7}$	$2.1 \cdot 10^{-8}$	0.80
	0.001	$10^{-5}$	$3.2 \cdot 10^{-8}$	$2.4 \cdot 10^{-9}$	0.84
Phase damping	0.030	$10^{-5}$	$8.2 \cdot 10^{-7}$	$4.0 \cdot 10^{-7}$	0.67
	0.010	$10^{-5}$	$3.2 \cdot 10^{-7}$	$3.7 \cdot 10^{-8}$	0.81
	0.008	$10^{-5}$	$2.4 \cdot 10^{-7}$	$2.4 \cdot 10^{-8}$	0.82
	0.005	$10^{-5}$	$1.2 \cdot 10^{-7}$	$9.4 \cdot 10^{-9}$	0.85
	0.003	$10^{-5}$	$5.1 \cdot 10^{-8}$	$3.6 \cdot 10^{-9}$	0.85
	0.001	$10^{-6}$	$6.9 \cdot 10^{-9}$	$5.0 \cdot 10^{-10}$	0.85

**Table 3.** Performance of the error mitigation algorithm evaluated using mean squared error (MSE) and tendency accuracy for different noise models and error rates.

## Conclusions

Understanding the binding affinity of a drug candidate can provide valuable insights into its potential efficacy and help to identify potential side effects. Additionally, predicting the binding affinity can help with the design of new molecules that bind more strongly to their target protein. This is especially important in the drug development process when dealing with new treatments that work on previously unexplored biological mechanisms. For this reason, designing efficient computational methods that can accurately predict the binding affinity between a molecule and a target protein is essential to speed up the drug discovery process.

Deep learning methods such as 3D CNNs provide promising results in this aspect since they learn directly from the atomic structure of the protein–ligand pair. Unfortunately, one of the biggest challenges of deep learning methods is the high complexity of the networks, which require learning millions of training parameters. This fact makes the training process long and costly, limiting the exploration of different network architectures. Quantum machine learning is a field that seeks to leverage the advantages of quantum computing to improve machine learning algorithms. Because of the exponential scaling of the Hilbert space, quantum computers can handle large and high-dimensional datasets and speed up machine learning algorithms.

In this paper, we present a hybrid quantum-classical 3D CNN, which reduces the complexity of the classical 3D CNN while maintaining an optimal prediction performance. With the proper design of quantum circuits, the hybrid CNN reduces the number of training parameters by 20%, which implies a reduction of training times of 20%–40%, depending on the hardware where the algorithm is executed. Apart from testing the performance of the algorithm in classical hardware, our work also proves the potential effectiveness of the method with noisy real hardware aided with a relevant error mitigation technique. Our results show that if the error probability is smaller than 0.01, a commonly-used error mitigation technique can accurately recover the noiseless outputs of the quantum circuit. All in all, this work shows how quantum machine learning offers the potential to reduce the complexity and long training times of classical neural networks by leveraging the advantages of quantum computing to handle large and high-dimensional datasets and speed up machine learning algorithms.

## Methods

This section provides an overview of the classical and quantum machine learning algorithms used in this study. It starts by discussing the PDBBind dataset and the per-processing methods used in our neural network models; the architecture of a classical 3D CNN is then described. All the processing algorithms and the architecture of the classical 3D CNN are the same as the ones in Ref.<sup>12</sup> to support a reproducible and comparable pipeline. Finally, the design of the hybrid quantum-classical CNN is described in detail.

## Data

The data used for this study is sourced from the PDBBind database<sup>22</sup> -a curated subset from the Protein Data Bank (PDB)-, which contains a collection of protein–ligand biomolecular complexes, manually collected from the associated publications. For each protein–ligand complex, the data files contain information about the 3D morphology, types of bonds between their constituent atoms, together with the protein–ligand binding affinity. Binding affinities were experimentally obtained by measuring the equilibrium dissociation constant between



protein–ligand,  $k_d$ , and the inhibition constant,  $k_I$ . Then, the binding affinity is defined as  $-\log(k_d/k_I)$ . Because of its completeness and extension, the PDBBind dataset has recently become a common benchmark for binding affinity prediction with both biophysics-based and machine learning methods<sup>17,26,27</sup>. The PDBBind dataset comes already split into two non-overlapping sets: the general and the refined set. The refined set is compiled in order to contain higher-quality complexes based on several filters regarding binding data (e.g complexes with  $IC_{50}$  but no  $k_i$  or  $k_d$  measurements), crystal structures (e.g low crystal resolution or missing fragments in the complex), as well as the nature of the complexes (e.g ligand–protein covalent bond binding). A segregated subset from the refined set, called *core set* provides a small, high-quality data collection for testing purposes.

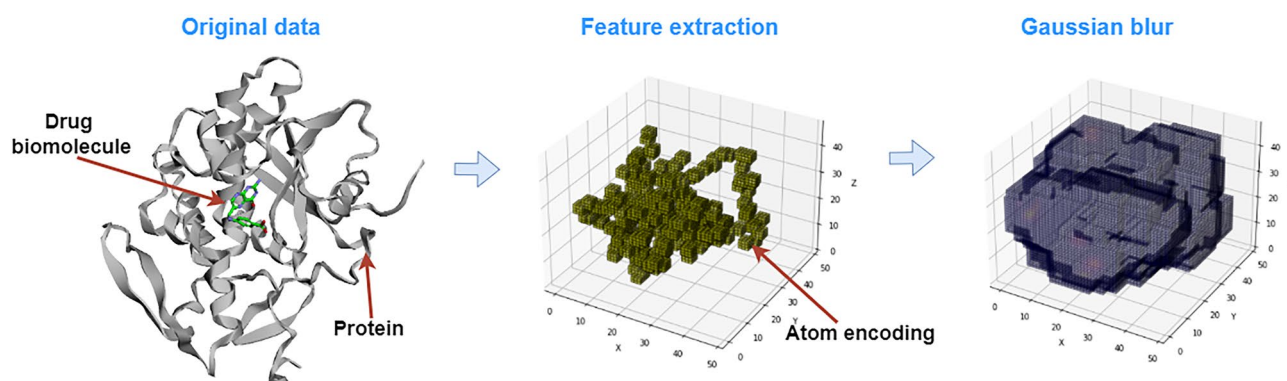
The 2020 version of the PDBBind dataset is used for this study. The general set (excluding the refined set) contains 14,127 complexes, while the refined set contains 5316 complexes. The core set is significantly smaller, with only 290 data samples.

### Data processing

In order to train classical and hybrid CNNs, the raw PDBBind data has to be transformed first into an appropriate input format for the convolutional layers. Before reshaping the data into the appropriate format, a common processing protocol was applied, following the same process as Ref.<sup>12,28</sup>. Hydrogens were added to all protein–ligand complexes according to each atom's valence. The partial charge of all the bonds is solved using UCSF Chimera<sup>29</sup> with the default settings. This protocol converts the PDB files to Mol2 files<sup>30</sup> (Molecular 2D files which store information about atoms, bonds, and molecular properties). A 3D spatial representation was then used to represent the features of the data. This method uses 3D volume grids to capture the atomic relationships in a voxelized space. That is, each data sample has size  $(C, N, N, N)$ , where  $N$  is the size of each dimension in space, and  $C$  is the number of features extracted from the protein–ligand pair. Here, we use a volume space size of 48Å and a voxel size of 1Å, so that  $N = 48$ . This size allows covering all the pocket regions without having too large input sizes for the CNN models. Having set the dimension of the space, the following 19 features were extracted from each protein–ligand pair ( $C = 19$ ):

- *Atom type*: One-hot encoding of the elements B, C, N, O, P, S, Se, halogen, or metal.
- *Atom hybridization*: Gives information about the number of  $\sigma$  and  $\pi$  bonds (i.e. geometry) connecting a particular atom to a neighboring atom. Takes values 1, 2 and 3 for  $sp$ ,  $sp^2$  and  $sp^3$  hybridizations, respectively.
- *Number of heavy atom bonds*: Heavy atoms are all but H.
- *Number of bonds with other heteroatoms*: Heteroatoms are those atoms different from H or C.
- *Structural properties*: one-hot encoding of hydrophobic, aromatic, acceptor, donor, and ring properties.
- *Partial charge*: Distribution of charge of an atom as a result of its chemical environment.
- *Molecule type*: Indicates whether it is a protein or a ligand atom ( $-1$  or  $1$ , respectively).

The feature extraction process was done with the OpenBabel tool (version 3.1.1.1)<sup>31</sup>. The Van der Waals radius was used to determine the size of each atom in the voxelized space. In this way, an atom could occupy one or more voxels depending on its Van der Waals radius. For atom overlaps, the features were added element-wise. The resulting 3D representations of the features resulted in sparse 3D matrices, which may make the training of neural networks harder since the input samples are too similar to each other, and to a zero-valued sample. Therefore, neural networks can have difficulties differentiating useful information from noise. For this reason, a Gaussian blur with amplitude  $\sigma = 1$  is applied to the voxelized features, populating the neighbouring atoms and thus reducing the number of zero-value voxels. Figure 6 shows a representation of the initial protein–ligand pair and the two main processing steps. Notice that these 3D volume representations are very high-dimensional since more than 2 million real numbers are needed to represent only one data sample. For this reason, large



**Figure 6.** Example of a data sample and further processing from the PDBBind dataset. The protein–ligand pair corresponds to the 1br6 sample from the refined set (left). The first step of the processing is feature extraction, where 19 features are voxelized in a 3D space (middle). Then, a Gaussian blur is applied to produce a dense representation of the data (right).

amounts of data samples and complex neural network models are needed to make accurate predictions without overfitting the training data.

The data processing is done independently for each of the datasets considered in this study. Apart from the general, refined and core sets, we further partitioned the general and refined sets into training and validation sets. This split is done to maintain the probability distribution of the binding affinities in both training and validation sets. For this reason, the binding affinities were separated into quintiles. Then for each quintile, we randomly selected 10% of the data for the validation set and kept the rest for the training set. In this way, we obtained training and validation sets for both the general and refined sets.

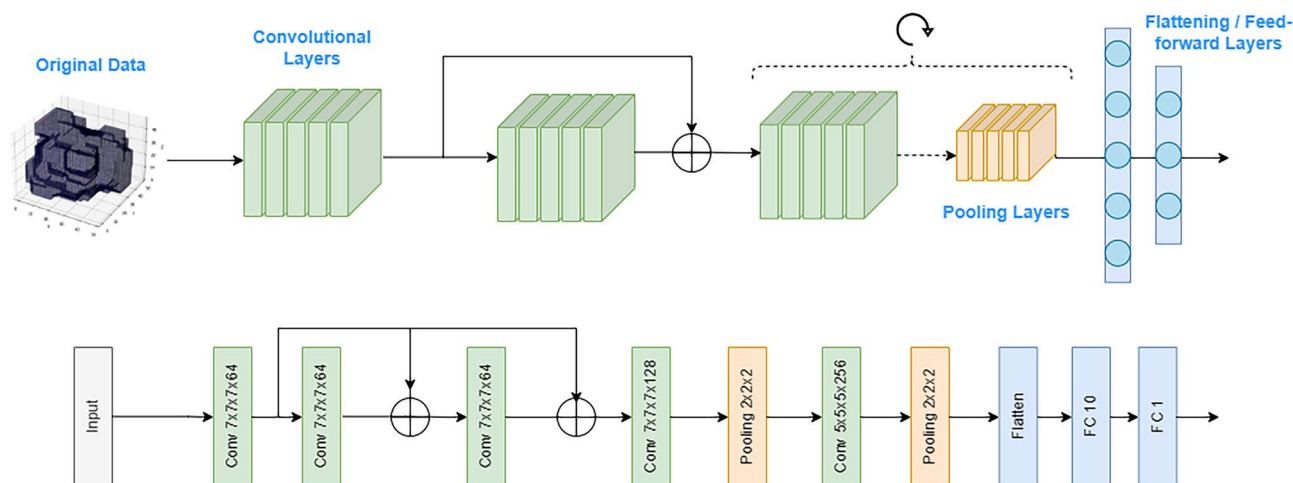
### Classical CNN

CNNs have provided very successful results in deep learning applications. This type of network is specialised in processing high-dimensional data in the form of spatial arrays, such as time series (in 1D), images (in 2D) or volumes (in 3D). The name stems from the fact that instead of general matrix multiplication, it employs a mathematical convolution in at least one of its layers. The output of the convolution is another array that represents some information that was present in the initial array in a very subtle way. In this way, a filter of a convolutional neural network is responsible for detecting one feature of the network input. The kernel matrices are free parameters that must be learned to perform the optimal feature extraction. The convolutional operation is followed by a nonlinear activation function which adds non-linearity to the system. Following the convolutional layers, a pooling layer is added to progressively reduce the spatial size of the array. After a series of convolutional and pooling layers, a flattening layer and some feed-forward layers are used to combine the extracted features and predict the final output.

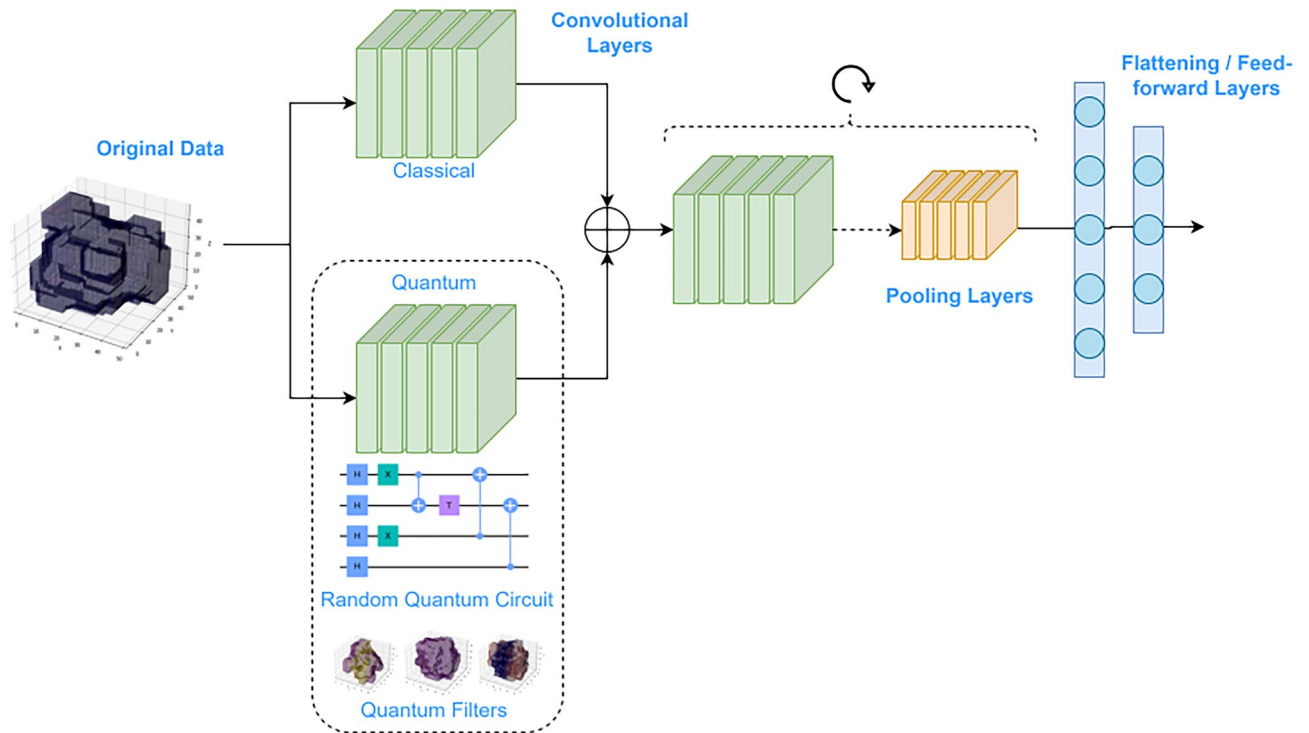
A representation of the layers of a 3D CNN is shown in Fig. 7 (Top). 3D CNNs have been used for multiple applications such as volume image segmentation<sup>32</sup>, medical imaging classification<sup>33</sup> and human action recognition<sup>34</sup>. A diagram of the 3D classical CNN used in this work is shown in Fig. 7 (Bottom). The architecture is the same as the one proposed in Ref.<sup>12</sup>, again for comparison purposes. The network contains five 3D convolutional layers, with 64,64,64,128 and 256 filters respectively. The kernel size is 7 for all the layers except for the last one, which has kernel 5. The CNN contains two residual connections, as proposed in ResNet<sup>35</sup>, which allow passing gradients to the next layers without a nonlinear activation function. Batch normalization is used after each convolutional layer, and we use Rectified Linear Unit (ReLU) as an activation function. The network contains two pooling layers and two fully-connected layers with 10 and 1 neurons respectively.

### Hybrid quantum-classical CNN

In this paper, we propose a hybrid (quantum-classical 3D) CNN, which is designed to reduce the complexity of the classical 3D CNN, while maintaining its prediction performance. Hybrid CNNs replace one or more convolutional layers with quantum convolutional layers<sup>36–38</sup>. That is, each classical convolutional filter is replaced by a quantum circuit, which acts as a quantum filter. These quantum circuits have significantly fewer training parameters than the classical convolutional layer, in order to reduce the overall complexity of the network. Each quantum circuit is divided into two blocks: the *data encoding*, which maps the input data into a quantum circuit, and the *quantum transformation*, where quantum operations are applied to retrieve information from the encoded data. In our case, our hybrid CNN replaces the first classical convolutional layer with a quantum convolutional layer. The final architecture of the hybrid CNN is depicted in Fig. 8. The processed protein–ligand data is fed to both a classical and a quantum convolutional layer. The outputs are aggregated by using a residual connection and then fed to the subsequent classical convolutional and pooling layers. The rest of the network is the same as its classical version. With this architecture, the first convolutional layer has been replaced by its quantum counterpart, while leaving the rest of the network unchanged.



**Figure 7.** Schematic representation of the components of a 3D convolutional neural network (top), and architecture of the 3D CNN used in this study (bottom), which was proposed in Ref. citeATOM.



**Figure 8.** Schematic representation of the hybrid quantum-classical 3D CNN. The original data is processed by both a classical convolutional layer and a quantum convolutional layer. The outputs of both layers are then aggregated. The result is then fed to a set of convolutional and pooling layers, following the same architecture as the classical 3D CNN.

*Data encoding*

The quantum convolutional layer aims to extract local features from the input data, just as the classical convolutional layer would. For this reason, we split the input data into  $(n \times n \times n)$ ,  $n \in \mathbb{N}$  blocks and process each block individually. Given a block  $B$ , the data encoding process converts  $B$  to a quantum state  $|B\rangle$ . Because of the high dimensionality of our data, we need to find a data encoding method that minimizes the number of qubits of the resulting quantum circuit. A suitable encoding should scale logarithmically with the dimension of the blocks. A popular data encoding mechanism that fulfills this property is called amplitude encoding<sup>39</sup>, which requires  $\lceil \log_2(n^3) \rceil$  qubits to encode a block. However, the amplitude encoding scheme normalizes each block independently to produce a normalized quantum state. Therefore, the different blocks of the data would have different normalization constants and would not be comparable with each other. For this reason, we decided to choose the Flexible Representation of Quantum Images<sup>40</sup> (FRQI) method, which normalizes the *whole* image before the encoding, avoiding this problem, and uses only  $\lceil \log_2(n^3) \rceil + 1$  qubits. FRQI was proposed to provide a normalized quantum state which encodes both the value (colour) of a pixel and its position in an image. Given an image with  $\theta = (\theta_0, \theta_1, \dots, \theta_{2^n-1})$  pixels, where the pixels have been normalized such that  $\theta_i \in [0, 2\pi), \forall i$ , the encoded state is given by Eq. (2).

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^n-1} (\cos \theta_i |0\rangle + \sin \theta_i |1\rangle) \otimes |i\rangle \tag{2}$$

where  $|i\rangle, i = 0, 1, \dots, 2^{2n}-1$  are the basis computational states, which is normalized since  $\| |I(\theta)\rangle \| = \frac{1}{2^n} \sqrt{\sum_{i=0}^{2^n-1} (\cos^2(\theta_i) + \sin^2(\theta_i))} = 1$ . For each  $\theta_i$ , the FRQI is composed of two parts:  $\cos \theta_i |0\rangle + \sin \theta_i |1\rangle$  encodes the color of the pixel, and  $|i\rangle$  encodes the position of the pixel in the image. As a simple example, a  $(2 \times 2)$  image and its representation are displayed as:

$$|I\rangle = \frac{1}{2} [ \begin{matrix} \theta_0, (00) & \theta_1, (01) \\ \theta_2, (10) & \theta_3, (11) \end{matrix} \otimes |00\rangle + (\cos \theta_1 |0\rangle + \sin \theta_1 |1\rangle) \otimes |01\rangle + (\cos \theta_2 |0\rangle + \sin \theta_2 |1\rangle) \otimes |10\rangle + (\cos \theta_3 |0\rangle + \sin \theta_3 |1\rangle) \otimes |11\rangle ]. \tag{3}$$

The number of qubits needed to construct the FRQI state increases logarithmically with the number of pixels (angles) of the image since the dimension of the computational basis increases exponentially with the number of qubits of the Hilbert space. In Ref.<sup>40</sup>, it is proven that the FRQI state can be implemented with simple quantum gates (Hadamard gates, CNOTs and  $R_y$  rotations). The number of quantum gates is polynomial with  $2^{2n}$ , the number of pixels of the image. Even though the FRQI was designed for 2D colour images, the generalization to 3D blocks is straightforward. Let  $B$  be a  $(n \times n \times n)$  block, with normalized values  $(\theta_0, \theta_1, \dots, \theta_{n^3-1}), \theta_i \in [0, 2\pi), \forall i$ . The FRQI state would then be given by

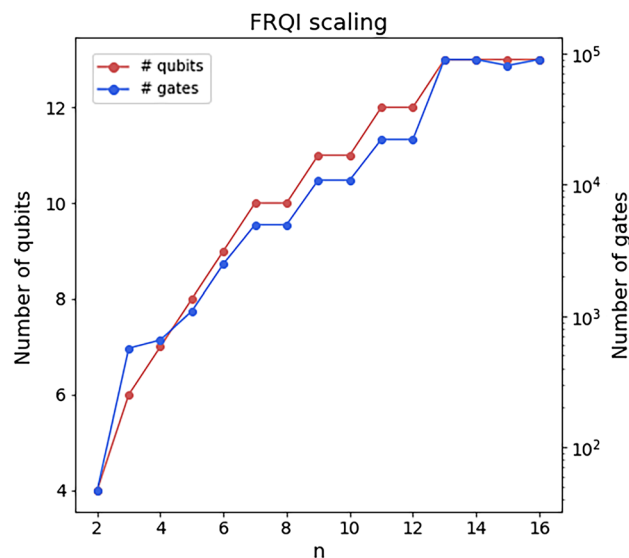
$$|B\rangle = \frac{1}{n^3} \sum_{i=0}^{n^3-1} (\cos \theta_i |0\rangle + \sin \theta_i |1\rangle) \otimes |i\rangle. \quad (4)$$

Notice that the only difference between Eqs. (2) and (4) is the number of angles of the quantum state. When  $n^3$  is a power of 2 (i.e.  $n^3 = 2^l, l \in \mathbb{N}$ ), the state in Eq. (4) has non-zero components in all the states of the computational basis. Therefore, choosing  $n^3$  as a power of 2 mostly exploits the use of the Hilbert space. For this reason, we set  $n = 4$  for our experiments. Figure 9 shows an example of the scaling of the number of qubits and the number of gates with the block size  $n$ . The number of qubits needed for the FRQI encoding is  $\lceil \log_2(n^3) \rceil + 1$ , so it scales logarithmically with the dimension of the block.

On the other hand, we have calculated the number of gates needed to implement the FRQI on a real quantum device. The number of gates depends on the values of the block  $\theta_i$ . If there are some angles with the same value, the quantum circuit can be compressed to reduce the number of gates. In Ref.<sup>40</sup>, the authors show how the FRQI quantum circuits can be simplified by minimizing boolean expressions. As an example of how the number of gates can scale with the block size, we have considered blocks from our data with the highest mean absolute sum, to ensure that we chose blocks with highly different angles. Figure 9 shows that the number of gates of a general FRQI encoding scales linearly with the dimension of the block,  $n^3$ . Recall that FRQI serves as an effective quantum encoding method tailored for handling high-dimensional arrays, such as images or volumes. Consequently, it is well-suited for quantum convolutional layers. However, when dealing with other types of neural networks, such as the widespread graph neural network<sup>12,41–43</sup>, different quantum circuit configurations need to be considered<sup>44,45</sup>.

#### Quantum transformation

After the data has been encoded in a quantum circuit, a set of quantum gates is applied to perform the quantum transformation, followed by a set of measurements that convert the data back to a classical representation (see the quantum circuit in Fig. 8). In a quantum convolutional layer, the quantum transformation is usually a parameterized quantum circuit (PQC), where the optimal parameters of the circuit need to be learned. However, because of the challenge represented by the high dimensionality of our data, many quantum complex circuits are needed to process a single data sample. By splitting the image in  $(4 \times 4 \times 4)$  blocks, 32832 quantum circuits are required to span the whole sample. The current hardware has limitations not only on the number of qubits and quantum gates but also on the number of quantum circuits that can be executed. For this reason, it is not possible to train our whole neural network model with the current hardware. Another option would be to run the PQC on quantum simulation. However, in this case, even though the quantum layer would still have fewer training



**Figure 9.** Example of scaling of the Flexible Representation of Quantum Images (FRQI). The number of qubits scales logarithmically, and the number of gates of the quantum circuit scales linearly with the dimension of the block  $n^3$ .

parameters than the classical convolutional layer, the training of quantum neural networks is not prepared to be run on GPUs, as compared with classical convolutional layers. For this reason, even though the hybrid model with PQCs has lower complexity, in our experiments the training time was longer than the classical CNN.

Another good alternative is using quantum reservoirs (QRs), an emerging approach in quantum machine learning, which has provided excellent results in multiple tasks<sup>46–48</sup>. It exploits the quantumness of a physical system to extract useful properties of the data that are then used to feed a machine learning model. In gate-based quantum computation, a QR is a *random* quantum circuit applied to an initial state, which encodes the input data, followed by measurements of local operators. These measurements are the features extracted by the model, which are then fed to a classical machine learning algorithm to predict the desired output. The main advantage of using QRs is the low complexity of the model, and thus, its easy training strategy. Instead of using PQC and finding its optimal parameters, QRs use carefully selected quantum systems with no training parameters to transform the input data. QRs have been used for temporal tasks (quantum reservoir computing<sup>47,49</sup>) and also to predict the excited properties of molecular data<sup>50,51</sup>.

In any case, the design of the random quantum circuit is crucial to determine the performance of the quantum machine learning model. Complex quantum circuits are the ones which better exploit the quantum properties of the system, and thus provide useful features for learning the target. In a recent work<sup>51</sup>, it was shown that the majorization principle<sup>52</sup> is a good indicator of both complexity<sup>53</sup> and performance<sup>51</sup> of a QR. That is, the QRs with higher complexity according to the majorization principle are the ones which give better results in the quantum machine learning tasks. In particular, seven families of quantum circuits, with different complexity, were used as QRs. For a given family, a quantum circuit is built by adding a fixed number of random quantum gates from such family. The G3={CNOT,H,T} family, where CNOT is the controlled-NOT gate, H stands for Hadamard, and T is the  $\pi/8$  phase gate, provided the best results when training the algorithm. Moreover, the performance of the QR increased with the number of gates of the circuit, until the performance reached its optimal value, and then it remained constant even if the number of gates increased. The role of noise in these computations was also taken into account in Ref.<sup>25</sup>

In this paper, the quantum transformation consists of a quantum circuit randomly generated with gates from the G3 family. Then, the qubits are measured on the computational basis, providing the output of the quantum convolutional layer. The hybrid CNN is trained with QRs with 20, 50, 100, 200, 300, 400, 500 and 600 quantum gates. In this way, we can evaluate how the depth of the QR influences the performance of the model. Figure 10 shows an example of the output of the quantum convolutional layer. We see that with a low number of gates, the quantum layer extracts simpler quantum features than with a higher number of gates.

Another widely used QR is the transverse-field Ising model<sup>46,47,50,54</sup>. In this case, the quantum circuit performs the time evolution of a quantum state under the random transverse-field Ising Hamiltonian

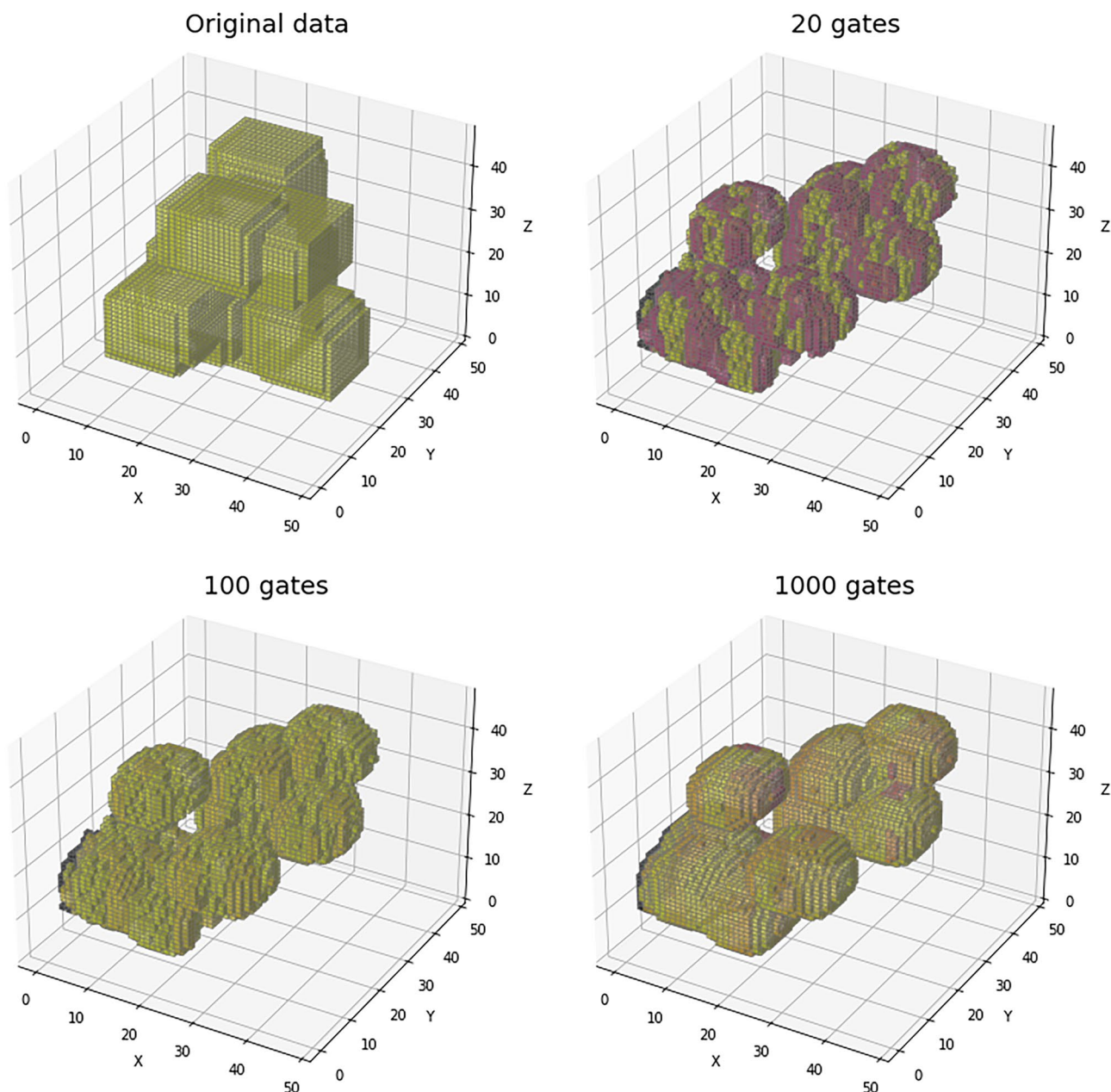
$$H_{\text{Ising}} = \sum_{i,j=0}^{N-1} J_{ij} Z_i Z_j + \sum_i^{N-1} h_i X_i, \quad (5)$$

where  $X_i$  and  $Z_j$  are Pauli operators acting on the site  $i, j$ -th qubit. The coefficients  $J_{ij}$  and  $h_i$  are chosen according to Ref.<sup>49</sup>, which provides a state-of-the-art method to select optimal parameters of the Ising model for quantum reservoir computing. In this case,  $J_{ij}$  are sampled from the uniform distribution  $U(-J_s/2, J_s/2)$  and  $h_i = h$  are constant. The optimal parameters fulfill  $h/J_s = 0.1$ . The system is evolved until time  $T = 10$ . We will compare the performance of the hybrid CNNs trained with QRs generated from the G3 family as well as the performance of the models with QRs generated from the Ising model. Since the current quantum computers have limited availability and high access queue times, which limit the number of iterative runs we can do for training, the hybrid CNNs are run using quantum simulation on classical hardware. The code has been optimized using Qiskit and PyTorch and adapted so that it could be trained on GPUs, just like the classical CNN.

### Error mitigation

One of the biggest challenges of the current quantum devices is the presence of noise. They perform noisy quantum operations with limited coherence time, which affects the performance of quantum algorithms. Even though the quantum circuits used for this study are run using quantum simulation, we have also evaluated the corresponding performance of the noisy quantum circuits using three different noise models for a small set of samples. The first noise model is the *amplitude damping channel*, which reproduces the effect of energy dissipation, that is, the loss of energy of a quantum state to its environment. The second noise model is described by the *phase damping channel*, which models the loss of quantum information without loss of energy. The last error model is described by the *depolarizing channel*. In this case, a Pauli error  $X, Y$  or  $Z$  occurs with the same probability  $p$ . For more information about the error models see Ref.<sup>21</sup>.

We perform here noisy simulations with error probabilities  $p = 0.03, 0.01, 0.008, 0.005, 0.003, 0.001$ . Error mitigation methods aim to reduce the noise of the outputs after the quantum algorithm has been executed. In this work, the *data regression error mitigation* (DRER) algorithm is used to mitigate the noise of the quantum circuits. The DRER algorithm trains a machine learning model to correct the errors of noisy quantum circuits. To obtain the training set, random quantum circuits with 300 gates sampled from the G3 family are executed with both noisy and noiseless simulations. Thus, the training set consists of pairs  $(X_i, y_i)$  where  $X_i$  contains the counts of the noisy distribution and  $y_i$  contains the counts of the noiseless distribution. In this case, the machine learning model we used is ridge regression, a regularized linear model which minimizes the mean squared error:



**Figure 10.** Example of an output of the quantum convolutional layer for different numbers of random gates in the G3 family, together with its input. The quantum convolutional layer is composed of a FRQI encoding layer followed by a quantum transformation generated with a random quantum circuit with different number of gates taken from the G3 family.

$$\text{MSE}_R = \frac{1}{N_s} \sum_{i=0}^{N_s} [W \cdot X_i - y_i]^2 + \alpha \|W\|^2 \quad (6)$$

where  $N_s$  is the number of samples in the training set,  $W$  is the matrix of the linear model,  $\alpha$  is the regularization parameter, and  $\|\cdot\|$  is the  $L^2$  norm. The DRER is trained with 1000 samples derived from the quantum layer's output, which are generated from data sourced from the refined set. Subsequently, its performance is evaluated using 500 noisy quantum circuits, also originating from the quantum layer's output, but this time using data from the core set. In this case, the 3D volumetric space is divided into blocks of size  $n = 8$ , leading to quantum circuits of 9 qubits and 300 gates. The DRER algorithm is suitable for this task since, once the machine learning model is trained, it can be used to mitigate multiple quantum circuits requiring very few classical computational resources. This makes it practical for use with large datasets.

### Data availability

The data used for this study is publicly available at <http://www.pdbbind.org.cn/22>.

## Code availability

The code used for this work can be found at Ingenii's open access library: <https://github.com/ingenii-solutions/ingenii-quantum-hybrid-networks/>.

Received: 7 August 2023; Accepted: 17 October 2023

Published online: 20 October 2023

## References

- Govind Kumar, V., Polasa, A., Agrawal, S., Kumar, T. K. & Moradi, M. Binding affinity estimation from restrained umbrella sampling simulations. *Nat. Comput. Sci.* **3**, 59–70. <https://doi.org/10.1038/s43588-022-00389-9> (2023).
- Liu, G. *et al.* Deep learning-guided discovery of an antibiotic targeting *Acinetobacter baumannii*. *Nat. Chem. Biol.* <https://doi.org/10.1038/s41589-023-01349-8> (2023).
- Paggi, J. M. *et al.* Leveraging nonstructural data to predict structures and affinities of protein–ligand complexes. *Proc. Nat. Acad. Sci.* **118**, e2112621118. <https://doi.org/10.1073/pnas.2112621118> (2021).
- Adcock, S. A. & McCammon, J. A. Molecular dynamics: Survey of methods for simulating the activity of proteins. *Chem. Rev.* **106**, 1589–1615. <https://doi.org/10.1021/cr040426m> (2006).
- Aldeghi, M., Heifetz, A., Bodkin, M. J., Knapp, S. & Biggin, P. C. Accurate calculation of the absolute free energy of binding for drug molecules. *Chem. Sci.* **7**, 207–218. <https://doi.org/10.1039/c5sc02678d> (2016).
- Towards a purely physics-based computational binding affinity estimation, *Nat. Comput. Sci.* **3**, 10–11 (2023) <https://doi.org/10.1038/s43588-023-00396-4>.
- Mey, A. S. *et al.* Best practices for alchemical free energy calculations, arXiv preprint [arXiv:2008.03067](https://arxiv.org/abs/2008.03067) (2020).
- Hahn, D. F. *et al.* Best practices for constructing, preparing, and evaluating protein–ligand binding affinity benchmarks, arXiv preprint [arXiv:2105.06222](https://arxiv.org/abs/2105.06222) (2021).
- Monticelli, L. & Tieleman, D. P. Force fields for classical molecular dynamics. In *Methods in Molecular Biology*, 197–213 (Humana Press, 2012) [https://doi.org/10.1007/978-1-62703-017-5\\_8](https://doi.org/10.1007/978-1-62703-017-5_8).
- Huang, S.-Y., Grinter, S. Z. & Zou, X. Scoring functions and their evaluation methods for protein–ligand docking: Recent advances and future directions. *Phys. Chem. Chem. Phys.* **12**, 12899–12908. <https://doi.org/10.1039/c0cp00151a> (2010).
- Shin, W.-H. & Seok, C. Galaxydock: Protein–ligand docking with flexible protein side-chains. *J. Chem. Inf. Model.* **52**, 3225–3232. <https://doi.org/10.1021/ci300342z> (2012).
- Jones, D. *et al.* Improved protein–ligand binding affinity prediction with structure-based deep fusion inference. *J. Chem. Inf. Model.* **61**, 1583–1592. <https://doi.org/10.1021/acs.jcim.0c01306> (2021).
- Wallach, I., Dzamba, M. & Heifets, A. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery, [ArXiv:abs/1510.02855](https://arxiv.org/abs/1510.02855) (2015).
- Kuzminykh, D. *et al.* 3d molecular representations based on the wave transform for convolutional neural networks. *Mol. Pharm.* **15**, 4378–4385. <https://doi.org/10.1021/acs.molpharmaceut.7b01134> (2018).
- Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. S. & De Fabritiis, G. DeepSite: Protein–binding site predictor using 3D-convolutional neural networks. *Bioinformatics* **33**, 3036–3042. <https://doi.org/10.1093/bioinformatics/btx350> (2017).
- Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J. & Koes, D. R. Protein–ligand scoring with convolutional neural networks. *J. Chem. Inf. Model.* **57**, 942–957. <https://doi.org/10.1021/acs.jcim.6b00740> (2017).
- Jiménez, J., Škalič, M., Martínez-Rosell, G. & De Fabritiis, G. Kdeep: Protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *J. Chem. Inf. Model.* **58**, 287–296. <https://doi.org/10.1021/acs.jcim.7b00650> (2018).
- Mohri, M., Rostamizadeh, A. & Talwalkar, A. *Foundations of Machine Learning* 2nd edn. (The MIT Press, 2018).
- Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**, 303–332. <https://doi.org/10.1137/S0036144598347011> (1999).
- Bharti, K. *et al.* Alán Aspuru-Guzik, Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **94**, 015004. <https://doi.org/10.1103/RevModPhys.94.015004> (2022).
- Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010). <https://doi.org/10.1017/CBO9780511976667>.
- wwPDB consortium. Protein Data Bank: The single global archive for 3D macromolecular structure data. *Nucleic Acids Res. Spec. Publ.* **47**, D520–D528. <https://doi.org/10.1093/nar/gky949> (2018).
- Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process.* **32**, 8024–8035 (2019).
- Czarnik, P., Arrasmith, A., Coles, P. J. & Cincio, L. Error mitigation with clifford quantum-circuit data. *Quantum* **5**, 592. <https://doi.org/10.22331/q-2021-11-26-592> (2021).
- Domingo, L., Carlo, G. & Borondo, F. Taking advantage of noise in quantum reservoir computing. *Sci. Rep.* **13**, 8790. <https://doi.org/10.1038/s41598-023-35461-5> (2023).
- Zhang, H., Liao, L., Saravanan, K. M., Yin, P. & Wei, Y. DeepBindRG: A deep learning based method for estimating effective protein–ligand affinity. *PeerJ* **7**, e7362. <https://doi.org/10.7717/peerj.7362> (2019).
- Wang, D. D., Chan, M.-T. & Yan, H. Structure-based protein–ligand interaction fingerprints for binding affinity prediction. *Comput. Struct. Biotechnol. J.* **19**, 6291–6300. <https://doi.org/10.1016/j.csbj.2021.11.018> (2021).
- Stepniewska-Dziubinska, M. M., Zielenkiewicz, P. & Siedlecki, P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics* **34**, 3666–3674. <https://doi.org/10.1093/bioinformatics/bty374> (2018).
- Pettersen, E. F. *et al.* Ucsf chimera—A visualization system for exploratory research and analysis. *J. Comput. Chem.* **25**, 1605–1612 (2004).
- Mol2 file format, Online <https://www.tripos.com/mol2/>.
- O’Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., & Hutchison, G. R. Open babel, Version 3.1.1.1 (2021) <http://openbabel.org/>.
- Hongya, L., Wang, H., Zhang, Q., Yoon, S. W. & Won, D. A 3d convolutional neural network for volumetric image semantic segmentation. *Procedia Manuf.* **39**, 422–428. <https://doi.org/10.1016/j.promfg.2020.01.386> (2019).
- Singh, S. 3d deep learning on medical images: A review. *Sensors* <https://doi.org/10.3390/s20185097> (2020).
- Arunnehr, J., Chamundeswari, G. & Bharathi, S. P. Human action recognition using 3d convolutional neural networks with 3d motion cuboids in surveillance videos. *Procedia Comput. Sci.* **133**, 471–477. <https://doi.org/10.1016/j.procs.2018.07.059> (2018).
- He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition, In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (2016) <https://doi.org/10.1109/CVPR.2016.90>.
- Oh, S., Choi, J. & Kim, J. A tutorial on quantum convolutional neural networks (qcnn), 236–239 (2020).
- Chen, S.Y.-C., Wei, T.-C., Zhang, C., Yu, H. & Yoo, S. Quantum convolutional neural networks for high energy physics data analysis. *Phys. Rev. Res.* **4**, 013231. <https://doi.org/10.1103/PhysRevResearch.4.013231> (2022).
- Henderson, M., Shakya, S., Pradhan, S. & Cook, T. Quantum convolutional neural networks: Powering image recognition with quantum circuits. *Quantum Mach. Intell.* **2**, 1–9. <https://doi.org/10.1007/s42484-020-00012-y> (2020).

39. Araujo, I. F., Park, D. K., Petruccione, F. & da Silva, A. J. A divide-and-conquer algorithm for quantum state preparation. *Sci. Rep.* **11**, 6329. <https://doi.org/10.1038/s41598-021-85474-1> (2021).
40. Le, P., Iliyasa, A., Dong, F. & Hirota, K. A flexible representation of quantum images for polynomial preparation, image compression and processing operations, quantum inf. *Quantum Inf. Process.* **10**, 63–84. <https://doi.org/10.1007/s11128-010-0177-y> (2011).
41. Wang, K., Zhou, R., Tang, J. & Li, M. Graphscoredata: Optimized graph neural network for protein–ligand binding affinity prediction. *Bioinformatics* **39**, btad340. <https://doi.org/10.1093/bioinformatics/btad340> (2023).
42. Yang, Z., Zhong, W., Lv, Q., Dong, T. & Chen, C.Y.-C. Geometric interaction graph neural network for predicting protein–ligand binding affinities from 3d structures (gign). *J. Phys. Chem. Lett.* **14**, 2020–2033 (2023).
43. Zhang, X. *et al.* Planet: A multi-objective graph neural network model for protein–ligand binding affinity prediction. *J. Chem. Inf. Model.* <https://doi.org/10.1021/acs.jcim.3c00253>, (2023).
44. Banerjee, S. *et al.* A hybrid quantum-classical fusion neural network to improve protein–ligand binding affinity predictions for drug discovery. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing* (2024) <https://doi.org/10.48550/arXiv.2309.03919>.
45. Zheng, J., Gao, Q. & Lü, Y. Quantum graph convolutional neural networks. In *2021 40th Chinese Control Conference (CCC)*, 6335–6340 (2021) <https://doi.org/10.23919/CCC52363.2021.9550372>.
46. Fujii, K. & Nakajima, K. Quantum reservoir computing: A reservoir approach toward quantum machine learning on near-term quantum devices. In *Reservoir Computing: Theory, Physical Implementations, and Applications* (eds Nakajima, K. & Fischer, I.) 423–450 (Springer Singapore, 2021).
47. Ghosh, S., Opala, A., Matuszewski, M., Paterek, T. & Liew, T. C. H. Quantum reservoir processing. *npj Quantum Inf.* **5**, 35 (2019).
48. Mujal, P. *et al.* Opportunities in quantum reservoir computing and extreme learning machines. *Adv. Quantum Technol.* **4**, 2100027 (2021).
49. Martínez-Peña, R., Giorgi, G. L., Nokkala, J., Soriano, M. C. & Zambrini, R. Dynamical phase transitions in quantum reservoir computing. *Phys. Rev. Lett.* **127**, 100502. <https://doi.org/10.1103/PhysRevLett.127.100502> (2021).
50. Kawai, H. & Nakagawa, Y. O. Predicting excited states from ground state wavefunction by supervised quantum machine learning. *Mach. Learn. Sci. Technol.* **1**, 045027 (2020).
51. Domingo, L., Carlo, G. & Borondo, F. Optimal quantum reservoir computing for the noisy intermediate-scale quantum era. *Phys. Rev. E* **106**, L043301. <https://doi.org/10.1103/PhysRevE.106.L043301> (2022).
52. Latorre, J. I. & Martín-Delgado, M. A. Majorization arrow in quantum-algorithm design. *Phys. Rev. A* **66**, 022305. <https://doi.org/10.1103/PhysRevA.66.022305> (2002).
53. Vallejos, R., de Melo, F. & Carlo, G. G. Principle of majorization: Application to random quantum circuits. *Phys. Rev. A* **104**, 012602 (2021).
54. Kutvonen, A., Fujii, K. & Sagawa, T. Optimizing a quantum reservoir computer for time series prediction. *Sci. Rep.* **10**, 14687 (2020).

## Acknowledgements

The project that gave rise to these results received the support of a fellowship from la Caixa Foundation (ID 100010434). The fellowship code is LCF/BQ/DR20/11790028. This work has also been partially supported by the Spanish Ministry of Science, Innovation and Universities, Gobierno de España, under Contracts No. PID2021-122711NB-C21

## Author contributions

L.D. Developed the idea and the theory, performed the calculations and analyzed the data, participated in discussions and interpretations of the results. C.J. and M.D. Developed the idea and the theory, participated in discussions and interpretations of the results. F.B. participated in discussions and interpretations of the results. All authors wrote and reviewed the manuscript.

## Competing interests

The authors declare no competing interests.

## Additional information

**Correspondence** and requests for materials should be addressed to L.D.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, 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 changes were made. 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/4.0/>.

© The Author(s) 2023