Machine Learning: Science and Technology

PAPER • OPEN ACCESS

A hybrid quantum regression model for the prediction of molecular atomization energies

To cite this article: Pranath Reddy and Aranya B Bhattacherjee 2021 *Mach. Learn.: Sci. Technol.* **2** 025019

MACHINE
LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
17 August 2020

REVISED
4 December 2020

ACCEPTED FOR PUBLICATION
17 December 2020

PUBLISHED
26 February 2021

PAPER

A hybrid quantum regression model for the prediction of molecular atomization energies

Pranath Reddy and Aranya B Bhattacherjee

Department of Physics, Birla Institute of Technology & Science, Pilani—Hyderabad Campus, Telangana, India

E-mail: aranyabhuti@hyderabad.bits-pilani.ac.in

Keywords: quantum machine learning, quantum regression model, atomization energies, quantum neural network

Abstract

Quantum machine learning is a relatively new research field that aims to combine the dramatic performance advantage offered by quantum computing and the ability of machine learning algorithms to learn complex distributions of high-dimensional data. The primary focus of this domain is the implementation of classical machine learning algorithms in the quantum mechanical domain and study of the speedup due to quantum parallelism, which could enable the development of novel techniques for solving problems such as quantum phase recognition and quantum error correction optimization. In this paper, we propose a hybrid quantum machine learning pipeline for predicting the atomization energies of various molecules using the nuclear charges and atomic positions of the constituent atoms. Firstly, we will be using a deep convolutional auto-encoder model for the feature extraction of data constructed from the eigenvalues and eigenvector centralities of the pairwise distance matrix calculated from atomic positions and the unrolled upper triangle of each Coulomb matrix calculated from nuclear charges, and we will then be using a quantum regression algorithm such as quantum linear regression, quantum radial basis function neural network and, a quantum neural network for estimating the atomization energy. The hybrid quantum neural network models do not seem to provide any speedup over their classical counterparts. Before implementing a quantum algorithm, we will also be using state-of-the-art classical machine learning and deep learning models such as XGBoost, multilayer perceptron, deep convolutional neural network, and a long short-term memory network to study the correlation between the extracted features and corresponding atomization energies of molecules.

1. Introduction

In recent years, machine learning and data-driven analysis methods have had a significant impact in various domains of science and technology, including medicine [1, 2], pharmaceutical [3], and chemical industries [4]. Efficient and accurate estimation of various fundamental chemical properties such as atomization energy of molecules is very crucial for the design and discovery of new compounds and drugs. Such tasks often require very computationally intensive simulations and calculations to derive empirical relations between various chemical parameters. However, machine learning algorithms and deep learning algorithms [5–7] have been proven to be very effective in learning these hidden patterns behind the complex distributions of high-dimensional data. This coupled with the speedup offered by the quantum computers, thanks to fundamental quantum mechanical concepts like superposition and entanglement [8] could revolutionize the field of computational drug design.

One of the main challenges of applying machine learning for the estimation of chemical properties of a molecule [9–11] is to efficiently encode the molecule as an input to the model. In this paper, we will be using the QM7 dataset [12, 13] that contains the Coulomb matrices, atomization energies, atomic charges and the Cartesian coordinates of atoms for a wide range of molecules. Similar to the previous works that have implemented machine learning methodologies using this dataset [14], we will try to map the atomization energies of the molecules using only the Coulomb matrices and the spatial locations of the atoms. In recent years, there has been an increasing interest in the study and development of novel quantum machine learning algorithms [15–20] and quantization of classical machine learning and deep learning models such as a quantum support vector machine [21–23], quantum neural networks [24–26], quantum convolutional neural networks (CNNs) [27], and quantum generative adversarial networks [28]. In this paper, our primary objective is to design and implement a hybrid quantum regression model where we use a classical deep learning model for feature extraction and a quantum regression algorithm for the prediction of atomization energies and study the feasibility of running these models on near term noisy intermediate-scale quantum computing (NISQ) systems.

We will be preparing our training dataset by unrolling only the upper triangular matrix of the Coulomb matrix since other elements of the matrix are redundant. We will also calculate a pairwise distance matrix using the Cartesian coordinates of the atoms and then calculate the eigenvalues and eigenvector centralities of the distance matrix. In section 2, we will use the concatenated arrays of unrolled Coulomb matrix, eigenvalues and eigenvector centralities for feature extraction to derive a compressed representation of the molecules. In section 3, we will be analyzing the correlation between the extracted features and the atomization energies of the molecules. Then in section 4, we will be looking at various quantum regression models before presenting the conclusions in section 5.

2. Methodology

The QM7 dataset is a part of the open-source database called quantum-machine that aims to facilitate the development of an efficient system for fast and accurate simulations of quantum-chemical systems from first principles. The QM7 dataset, in particular, is a dataset aimed toward the prediction of atomization energies from molecular geometry and is a subset of GDB-13, a database of 1 billion stable and synthetically accessible organic molecules. The dataset consists of 7165 molecules that are composed of 23 different atoms including heavy atoms like C, N, O, and S. The first component of the dataset is the Coulomb matrix which is a symmetric matrix of size 23 × 23 calculated from the nuclear charges of the atoms that constitute a particular molecule.

The Coulomb matrix is calculated as follows:

$$Cij = \begin{cases} 0.5 \times Z_i^{2.4}, & \forall i = j \\ \frac{Z_i \times Z_j}{||Ri - Rj||}, & \forall i \neq j \end{cases} \tag{1}$$

where Z_i is the nuclear charge of an atom i that is located at a position R_i. To avoid redundancy, we will be taking the upper triangular matrix of the Coulomb matrix, including the diagonal and unrolling it to map a matrix of size 23 × 23 to an array of size 1 × 276. The Coulomb matrix is invariant to translation and rotation of the molecules. The dataset also consists of Cartesian coordinates of the atoms from which we have calculated the pairwise interatomic distance matrix. We have then unrolled the distance matrix by taking the upper triangular matrix of the matrix, excluding the diagonal matrix mapping a matrix of size 23 × 23 to an array of size 1 × 253. The pairwise interatomic distance matrix of a molecule is a two-dimensional square matrix containing the distances taken pairwise, between the atoms of a specific molecule .We have plotted the distribution of average interatomic distance of the molecules to observe the distribution, as shown in figure 1.

We then calculate the eigenvalues and eigenvector centralities of the distance matrix to account for the intensity and direction of the spread of the data, which are essential in understanding the influence of an individual atom on the entire molecule. Considering the pairwise distance matrix to be an adjacency matrix ($D = d_{(v,t)}$, $d_{(v,t)} = 1$ if v is linked to t) of a graph $G(V, E)$ with $|V|$ vertices, we can calculate the eigenvalues and eigenvector centralities. The relative centrality score of vertex v (c_v) can be defined as

$$c_v = 1/\lambda \times \sum_{t \in M(v)} (c_t) = 1/\lambda \times \sum_{t \in G} (d_{(v,t)} \times c_t) \tag{2}$$

where $M(v)$ is the set of neighbours of v, λ is the greatest eigenvalue of D [29], and c_t is the relative centrality score of vertex $t \in G$. Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors, and it is a measure of a node's importance. The calculation of Eigenvector centrality is iterative and we use a normalized vector with equal values as a starting vector for the centrality scores. The size of eigenvalues and eigenvector centralities are 1 × 23 and 1 × 23, respectively. We then concatenate the flattened Coulomb matrix, pairwise distance matrix, eigenvalues, and eigenvector centralities, which gives a final array of size 1 × 575 for an individual molecule and a dataset of size 7165 × 575.

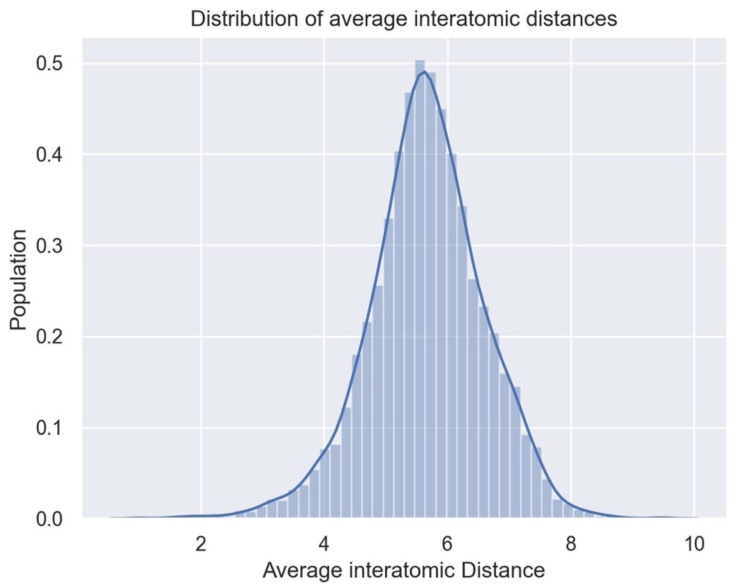

Figure 1. Distribution of average interatomic distances of the molecules.

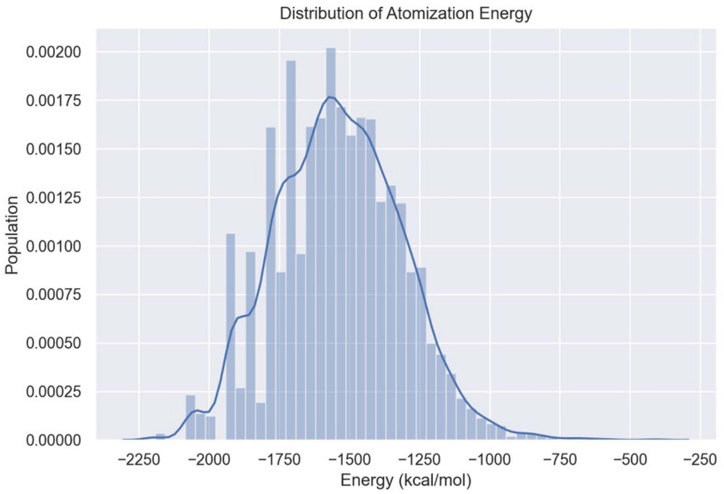

Figure 2. Distribution of atomization energies of the molecules.

The atomization energies of the molecules are taken as the labels for our regression model. The atomization energies are denoted in kcal mol^{-1}, and their values range from -404.88 to -2192 kcal mol^{-1}. We have also plotted the atomization energies of the molecules to observe the distribution, as seen in figure 2.

We will now be using a deep convolutional auto-encoder for feature extraction to derive a compressed representation of our dataset since amplitude encoding of our preprocessed QM7 dataset requires a large number of qubits even though amplitude encoding is the most qubit efficient form of encoding. Autoencoder models consist of an encoder network and a decoder network. The encoder network learns to map the input samples to a latent vector and the decoder network learns to reconstruct the input from the latent dimension. Deep convolutional auto-encoders [30] have been used for feature extraction in various video and signal processing and analysis problems and have proven to be very effective in reconstructing the complex distributions of data [31, 32]. The architecture of our Deep convolutional auto-encoder is shown in figure 3.

We have used 64, 32, 16, and 8 filters (kernels) of size (18,1) in the convolution layers and a pool size of (2,1) for the average pooling layers. Similarly, we have used 16, 32, and 64 filters (kernels) of size (18,1) in the transposed convolutional layers and a scale factor of (2,1) in the upsampling layers. We have used a linear activation function throughout the network and trained the model using a stochastic gradient descent optimizer [33–35] with a mean squared error loss function at a learning rate of 0.001. We have trained the model for 1000 epochs with a batch size of 100. The model has been implemented using the Keras library

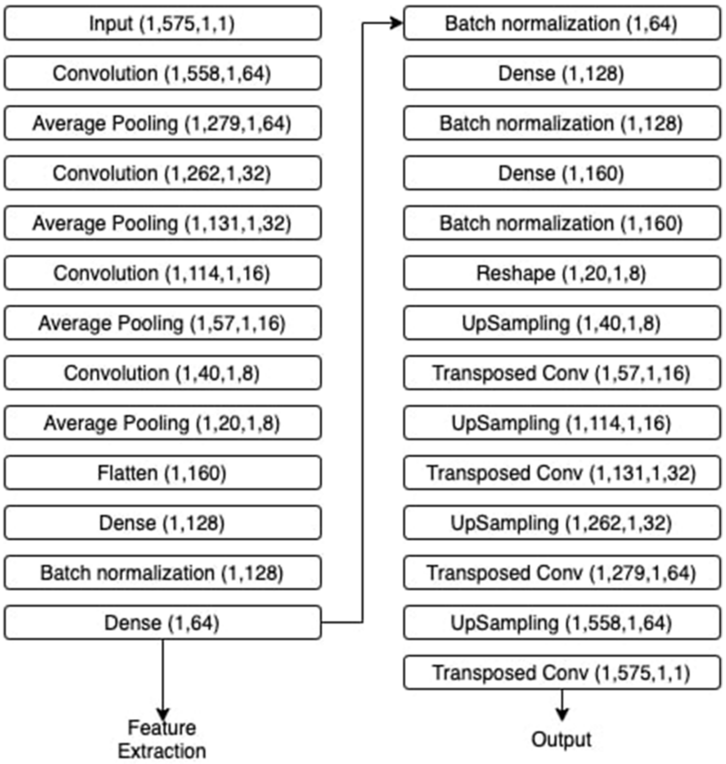
Input (1,575,1,1)
Convolution (1,558,1,64)
Average Pooling (1,279,1,64)
Convolution (1,262,1,32)
Average Pooling (1,131,1,32)
Convolution (1,114,1,16)
Average Pooling (1,57,1,16)
Convolution (1,40,1,8)
Average Pooling (1,20,1,8)
Flatten (1,160)
Dense (1,128)
Batch normalization (1,128)
Dense (1,64)
Feature Extraction
Batch normalization (1,64)
Dense (1,128)
Batch normalization (1,128)
Dense (1,160)
Batch normalization (1,160)
Reshape (1,20,1,8)
UpSampling (1,40,1,8)
Transposed Conv (1,57,1,16)
UpSampling (1,114,1,16)
Transposed Conv (1,131,1,32)
UpSampling (1,262,1,32)
Transposed Conv (1,279,1,64)
UpSampling (1,558,1,64)
Transposed Conv (1,575,1,1)
Output

Figure 3. Architecture of the deep convolutional auto-encoder model. Each block represents a layer of the model with information about the type of layer and the output shape. The output shape for the convolution and pooling layers is in NHWC format (batch size, spatial dimensions, channels), in this case, (1, length, 1 channels).

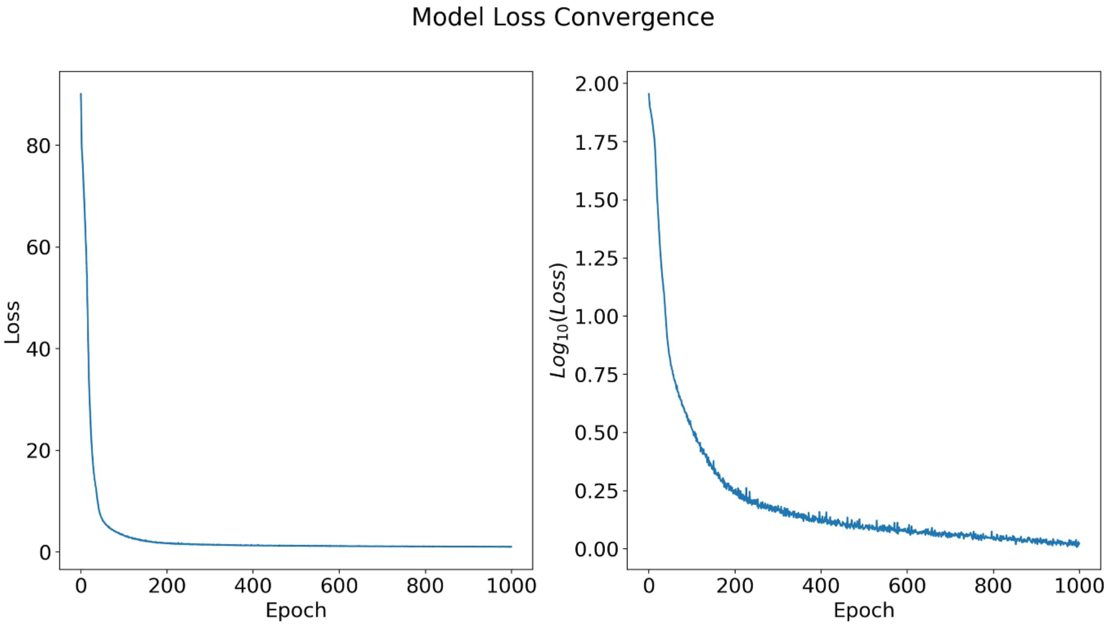

Figure 4. Convergence plot of training loss of the deep convolutional auto-encoder model.

with Tensorflow backend and trained on a Google Colaboratory notebook with GPU (Nvidia Tesla K80) acceleration. The model loss convergence is shown in figure 4.

We have taken the features from the bottleneck layer of the model, which gives us a feature vector of dimension 1 × 64 for each molecule. Hence we have mapped out a dataset of size 7165 × 575 to a compressed representation of size 7165 × 64. We have used metrics like correlation matrix, 2-component kernel PCA (principal component analysis) which is an extension of PCA that uses a kernel function (in our case, a radial basis function) to map the dataset into a lower-dimensional space, and a t-distributed stochastic neighbor

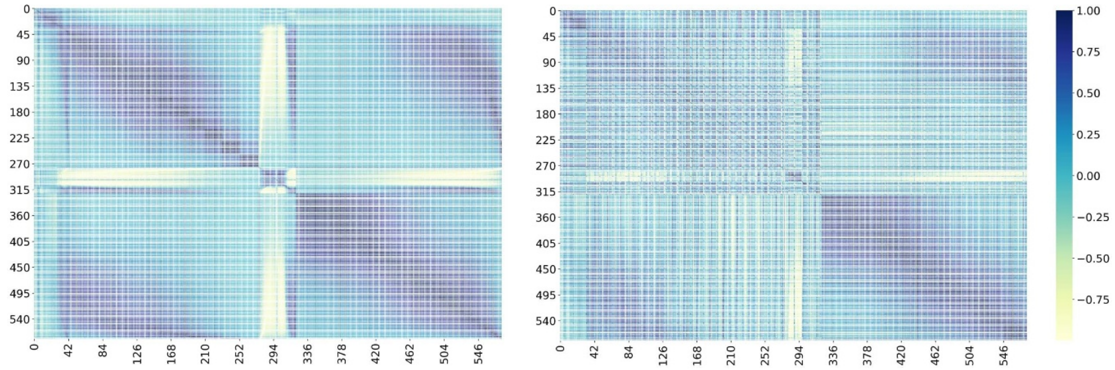

Figure 5. Correlation matrix of training data (left) and the correlation matrix of reconstructed data (right). The correlation matrix shown here is an array containing all the Pearson correlation coefficients for different variables in our data. The upper left and lower right quadrants correspond to the auto-correlation values and the lower left and upper right quadrants correspond to the cross-covariance values.

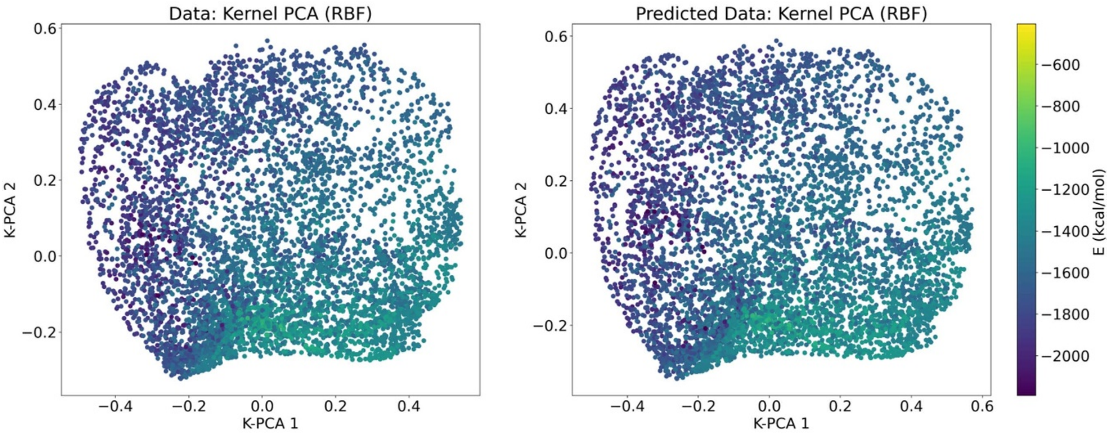

Figure 6. K-PCA result of training data (left) and K-PCA result for reconstructed data (right). The first two principal components (k-PCA 1, k-PCA2) are plotted and a colormap is added to observe the clustering of values.

embedding to compare the actual data to the reconstructed data, and evaluate the performance of the model. Similar to kernel PCA, t-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimensionality reduction technique. t-SNE minimizes the KL-divergence [36] between the distributions of the high-dimensional data and the mapped low-dimensional data.

The reconstructed results closely resemble the actual data, and the slight discrepancies in the Correlation matrix and K-PCA results can be attributed to the noise-induced in the reconstructed samples due to the reconstruction loss of the network. The results are shown in figures 5, 6, and 7.

3. Classical regression methods

Before applying quantum regression methods for predicting the atomization energy of the molecules using the extracted features, we will be looking at some classical regression methods such as XGBoost [37], multilayer perceptron (MLP) [38], deep convolutional neural network (CNN) [39] and a bidirectional long short-term memory recurrent neural network (LSTM)-RNN [41] (An implementation of a recurrent neural network using LSTM modules) model to study and confirm the correlation between the extracted features and corresponding atomization energies of molecules.

XGBoost is an open-source framework for gradient-boosted decision trees. An MLP is one of the most commonly used models that belong to the class of feedforward ANNs. Each neuron in one Layer of the MLP is connected to the neurons of the next layer via a non-linear activation function. An optimization algorithm called back-propagation makes MLPs very effective in learning and generalizing the distributions of data. Recurrent neural networks are widely used in various machine learning problems such as speech recognition [42], financial data analysis [43], time series forecasting [44], and time series classification [45]. The main feature that makes RNN highly versatile when working with them is their ability to learn sequences

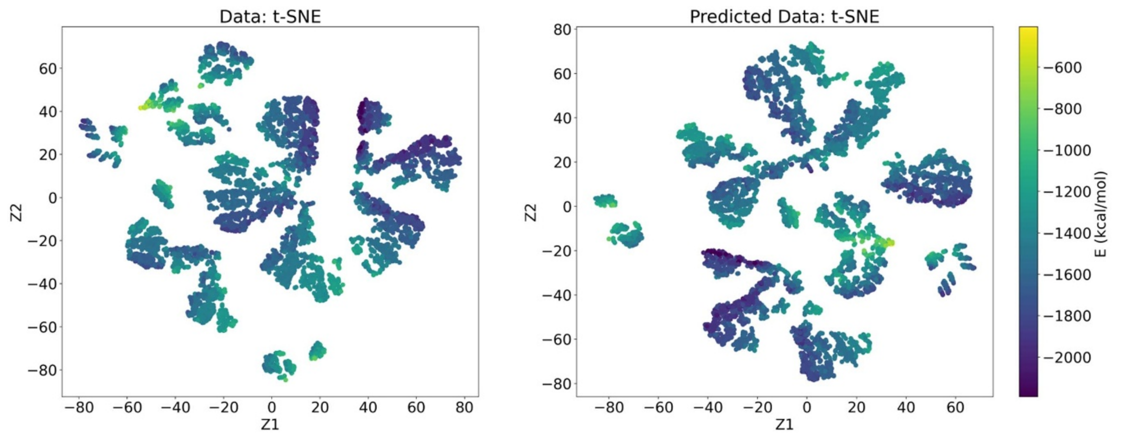

Figure 7. t-SNE result of training data (left) and t-SNE result for reconstructed data (right). The scatter plot shows the clusters on the low-dimensional representation of the high-dimensional space.

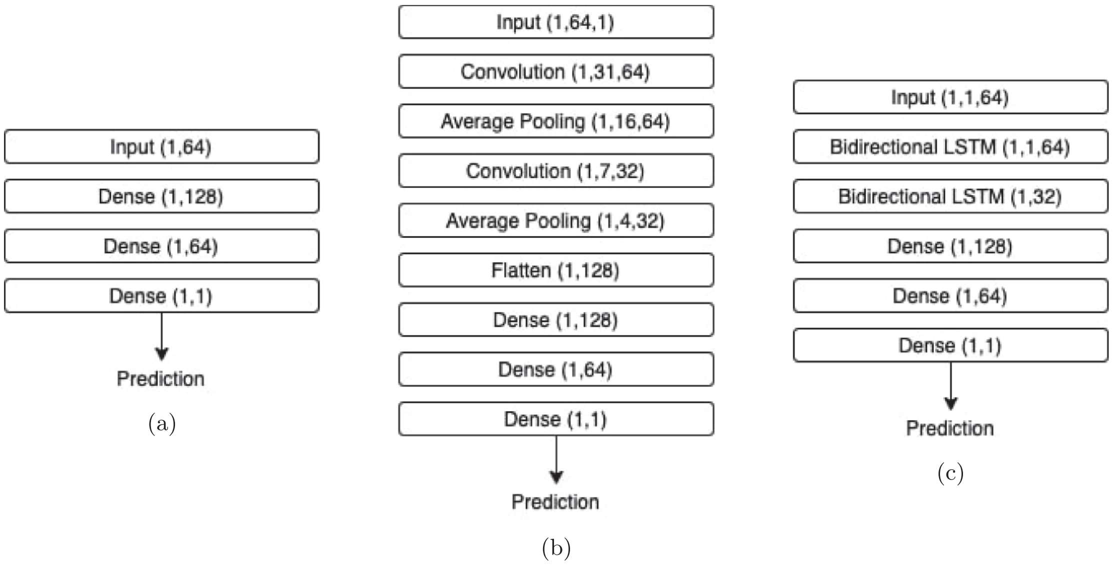

Figure 8. Architectures of MLP model (a), deep CNN model (b), and LSTM model (c). Each block represents a layer of the model with information about the type of layer and the output shape.

in the data. CNN is primarily used for image and video data, but we can use a deep CNN model for our regression problem due to its ability to learn non-linear hierarchical structures and lower computational costs. The architectures of the models are shown in figure 8.

For the XGBoost model, we have used mean absolute error (MAE) for the evaluation metric, and a learning rate of 0.01 and the maximum depth of a tree set to 6. For the MLP, CNN, and LSTM models, we have used a rectified linear unit activation function in the hidden layers and a linear activation function in the output layer. We have used the ADAM optimizer [46] with a learning rate of 1×10^{-4} and decay of 1×10^{-5}. All three models have been trained for 10 000 epochs with a batch size of 128. The models have been implemented using the Keras library with Tensorflow backend and trained on a Google Colaboratory notebook with GPU (Nvidia Tesla K80) acceleration. We have used hold-out cross-validation with an 80-20 split to evaluate the performance of our models, and the results are presented in table 2.

The CNN model performs the best and the results of the MLP and LSTM model are comparable. The results of the LSTM model shows us that there is not any specific time dependence or sequential features in the input data. The results of the XGBoost model are not as good as what we see from the deep-learning-based models which is an expected behavior when studying high dimensional data. The MAE and the root mean squared error (RMSE) values are fairly low but are still far from current benchmark values of around 3 kcal mol^{-1}, this is due to the fact that we are using the extracted features for regression which is a

Table 1. Results of classical regression models.

Metric	XGBoost	MLP	LSTM	CNN
MAE (kcal mol^{-1})	20.31 307	16.78 842	18.33 729	12.28 713
RMSE (kcal mol^{-1})	26.91 781	23.77 603	25.64 239	16.56 891
R2	0.98 537	0.98 787	0.98 705	0.99 278
Pearson correlation	0.99 278	0.99 392	0.99 351	0.99 627

Table 2. Results of linear regression and quantum linear regression models.

Metric	Linear regression	Quantum linear regression
MAE (kcal mol^{-1})	74.25 348	74.24 690
RMSE (kcal mol^{-1})	93.55 287	93.54 265
R2	0.81 232	0.81 236
Pearson correlation	0.91 873	0.91 882

very compressed representation of the actual dataset. The coefficient of determination (R2) and Pearson correlation values of all the models are close to 1, which indicates that the extracted features are indeed correlated to the atomization energies of the molecules and can be used for designing and testing quantum regression models.

4. Quantum regression methods

4.1. Quantum linear regression

When studying large volume datasets, current machine learning models often encounter a performance bottleneck when trained on classical systems [47], but thanks to the parallel computing power of quantum systems, we can solve this problem by developing quantum analogous models of classical machine learning algorithms. In this section, we will be implementing and comparing the results of a common curve fitting model called linear regression and quantum linear regression [48]. The primary advantage of the quantum linear regression method is the logarithmic computation time when computed on a quantum processor using quantum random access memory (qRAM) as opposed to the linear computation time of classical Vectorized Linear regression. Ewin Tang's work [49, 50], which involves l^2-norm sampling, demonstrated that for manipulating non-sparse low-rank matrices, quantum algorithms may be dequantized to reduce the exponential speedup to polynomial advantage. But, since we are dealing with a high-rank matrix, the quantum linear regression algorithm provides a significant performance advantage over its classical counterpart [51, 52]. We will be using the extracted features for testing quantum regression models. We will be implementing the quantum linear regression model by encoding our features vectors as quantum states using sparse matrix representations. The feature vectors are concatenated into a matrix X,

$$X \in R^{N\times M} : (X(1), X(2), X(3), \ldots .X(N))^T, \tag{3}$$

with each row of the matrix as a feature vector and the atomization energies will be stored in a vector Y $\in R^{N^*1}$. We will be using hold out cross-validation to split the data into 80% training and 20% testing/validation. Our implementation of quantum linear regression will be based on classical Vectorization-based' linear regression where the weight matrix W is calculated by calculating the gradient of least squares cost. We will also be adding Gaussian noise (Δ) of mean 0 and a standard deviation of 0.1 to replicate the behavior of NISQ systems. The linear algebra operations have been implemented using a python framework for numerical simulations of quantum systems called QuTip [53]. The results are presented in table 2. The results of the linear regression model and the quantum regression model are both very comparable and we can see that the quantum regression model performs slightly better in spite of the addition of Gaussian noise which could be due to the sparse encoding of the input vectors:

$$W = ((X \times X^{\dagger})^{-1} \times X^{\dagger}) \times Y + \Delta, \tag{4}$$

$$\hat{Y} = \hat{X} \times W, \tag{5}$$

where $\hat{Y}$ are the predicted atomization energies and $\hat{X}$ is the testing/validation dataset.

Table 3. Results of RBFNN and Q-RBFNN models.

Metric	RBFNN	Q-RBFNN
MAE (kcal mol^{-1})	33.43 229	43.64 339
RMSE (kcal mol^{-1})	44.13 469	53.95 243
R2	0.95 823	0.93 375
Pearson correlation	0.97 891	0.96 692

4.2. Quantum radial basis function neural network (Q-RBFNN)

We will now be implementing a Q-RBFNN by amplitude encoding of the feature vectors. In reference [54], the authors proposed a different version of a quantum RBFN, intending to solve classification problems and implemented by defining the state of the weight as a tensor product of single-qubit states. Instead, in our article, we propose to solve a regression problem and implement it by defining a new quantum equivalent activation function based on the Hilbert–Schmidt norm. We will be encoding the feature vector as amplitudes of standard basis based on the qRAM mapping [55, 56]. This encoding will exponentially reduce the computation time for finding the centroids of the hidden layer of radial basis function neural network (RBFNN). In classical RBFNN, we find the centroids in the clustering step of the algorithm by calculating the Euclidian distance of the vectors by assuming them as Cartesian coordinates in a high dimensional space, but in Q-RBFNN, we will be using the trace distance of the density matrices of our amplitude encoded states. We will also be adding multivariate Gaussian noise (Δ) of mean 0 and covariance 0.1 to replicate the behavior of NISQ systems. We will be using the Hilbert–Schmidt norm of the difference between the centroids and input states as the radial basis function of our Q-RBFNN model. Hilbert Schmidt norm is a matrix norm operator, it is the square root of the trace of the inner product of two states. Since we are using a specific radial basis function, the Q-RBFNN algorithm can be interpreted as a specific instance of the RBFNN algorithm. The results have been presented in table 3. We can see a percentage difference of 26.4963 in the MAE score and a percentage difference of 20.0184 in the RMSE score. While there is a significant difference in the performance of the classical and quantum RBFNN models, the R2 score and the Pearson correlation value of the Q-RBFNN model signifies that the model is capable of learning the distribution of the input data.

Algorithm 1: Classical RBFNN.

```
1. Perform classical K-Means clustering for set numbers of neurons in the hidden layer
    (a) The number of clusters is equal to the number of hidden neurons
    (b) Let the centroids be $\mu^K$ for K clusters
2. Evaluate the number of instances or feature vectors which fall under cluster ‘k’ and denoted as $X_i^k$
3. Construct the Hidden layer matrix H as:
        $H = [\phi(||X_i^k - \mu^k||)]_{i=1,k=1}^{m,K}$        (6)
    where $\phi$ is the radial basis function of choice.
4. Construct the weight matrix and then calculate the predicted outputs.
        $W = H^{-1} \times Y$        (7)
        $\hat{Y} = \hat{X} \times W$        (8)
    where $\hat{Y}$ are the predicted atomization energies and $\hat{X}$ is the testing/validation dataset.
```

4.3. Hybrid quantum neural network and QFT-based hybrid QNN

We will now be implementing a hybrid quantum neural network using the variational method. The feedforward network will be built using classical convolutional and dense layers and variational circuits with parametrized rotation gates, and the optimization and updating of the parameters corresponding to the network gates will be done using classical methods. The input feature vectors will be encoded into quantum states using amplitude encoding and, each quantum state corresponding to the feature vector will be used as the input to the classical network and the output vector of the classical layer before the quantum circuit will be used as the input state to the quantum circuit. The quantum circuit is comprised of a series of parametrized rotation gates along x, y and z-axis and the measurements of the circuit are mapped to a single output value using fully connected layers, and we calculate the cost between this output value and the label to

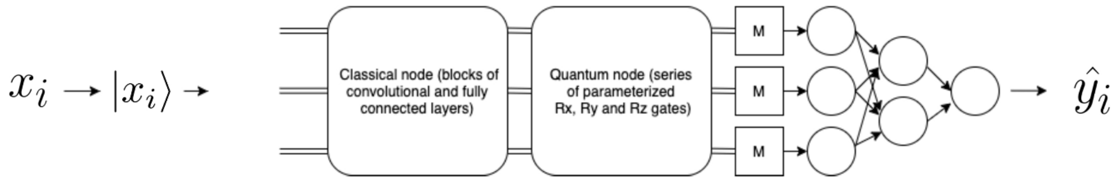
$x_i \rightarrow |x_i\rangle \rightarrow$
Classical node (blocks of convolutional and fully connected layers)
Quantum node (series of parameterized Rx, Ry and Rz gates)
M
M
M
$\hat{y}_i$

Figure 9. General representation of a hybrid QNN model.

update the parameters of the circuit based on a hybrid quantum–classical optimization method using the parameter-shift rule and gate decomposition as described in the paper by Crooks [57]. The atomization energies of the samples have been rescaled to the range [0, 1], and we will be primarily focusing on studying the ability of our hybrid QNN model to learn the distribution of the data, and for this, we have chosen training convergence as the parameter of interest. We have used IBM's Qiskit library with the QASM simulator backend for building the quantum circuit and PyTorch library for implementing the classical neural network part and for the optimization of the entire model. A general representation of the models are shown in figures 9 and 10.

Algorithm 2: Q-RBFNN.

1. Encode the input vectors as quantum states by amplitude encoding them as amplitudes of standard basis.

   (a) For an input vector of dimension m, the encoded quantum state is (can be done using m qubits):

$$|x_i\rangle = \frac{\sum x_i^m \times |m\rangle}{||x_i||} \tag{9}$$

2. Perform clustering based on the trace distance of density matrices and let the centroids be $|\mu^K\rangle$ for K clusters.

   (a) The trace distance of density matrices if denoted as:

$$\delta(\rho_c, \rho_x) = 1/2 \times (tr(|\rho_c - \rho_x|)) \tag{10}$$

3. Calculate the Hidden layer matrix using the Hilbert–Schmidt norm of the difference between the centroids and input states:

$$H = [\sqrt{tr((|x^K\rangle - |\mu^K\rangle)^\dagger \times (|x^K\rangle - |\mu^K\rangle))}]_{i=1,k=1}^{m,k} + \Delta \tag{11}$$

4. Construct the weight matrix and then calculate the predicted outputs.

$$W = H^{-1} \times Y \tag{12}$$

$$\hat{Y} = \hat{X} \times W \tag{13}$$

   where Ŷ are the predicted atomization energies and X̂ is the testing/validation dataset.

Additionally, we will also be studying a quantum Fourier transform (QFT)-based hybrid QNN model. Fourier transform is a way of representing temporal data in the frequency space. Fourier transform is one of the most frequently used operations in signal processing and also machine learning. Analyzing the data in frequency space could provide new information about the distribution of the data and can also help ML models learn the representations of the data more efficiently. The QFT is analogous to classical discrete Fourier transform, but it is applied over the amplitudes of a given wavefunction, converting states in computational basis to Fourier basis. QFT circuit can be implemented using a series of controlled phase rotations and SWAP gates. The QFT circuit transforms the bases with the help of a series of H-gates that transform the Z-basis states $|0\rangle$ and $|1\rangle$ to the X-basis states $|+\rangle$ and $|-\rangle$. They similarly map all multi-qubit states in the computational basis to their corresponding Fourier basis states. A six qubit QFT circuit is shown in figure 11. We will be using the amplitude encoded feature vectors as the input to a QFT circuit (a 64-dimensional input state is passed to a six-Qubit QFT circuit), and the output states of the QFT circuit will be used as the inputs to a hybrid QNN. For a classical k-dimensional vector, the quantum state encoding requires $\log_2(k)$ qubits, and in general, the time complexity of quantum state preparation is $O(\log_2(N))$ [56],

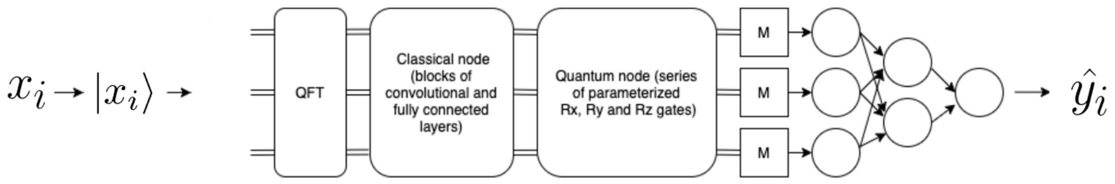
$x_i \rightarrow |x_i\rangle \rightarrow$
QFT
Classical node (blocks of convolutional and fully connected layers)
Quantum node (series of parameterized Rx, Ry and Rz gates)
M
M
M
$\rightarrow \hat{y}_i$

Figure 10. General representation of a QFT-based hybrid QNN Model.

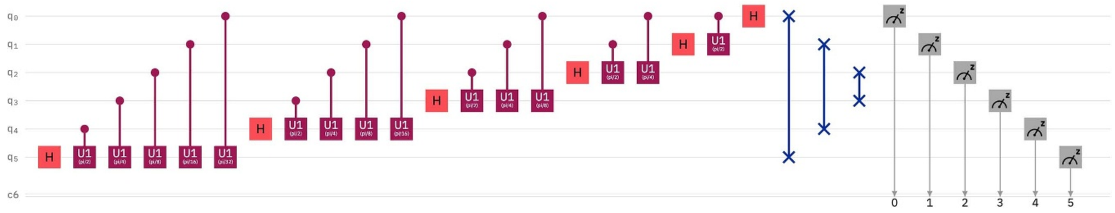

Figure 11. A six Qubit QFT circuit visualized using the Qiskit framework.

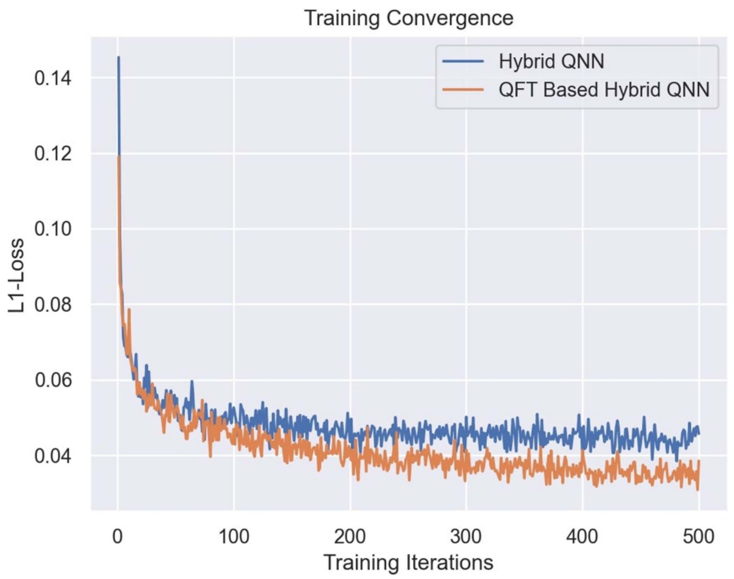

Figure 12. Training Convergence of the hybrid QNN circuit model and QFT-based hybrid QNN circuit model.

where N is the dimension of the vector k. The complexity of arbitrary quantum state preparation is $O(N)$. We hypothesize that the Fourier basis states could help our QNN model achieve better results compared to a standard hybrid QNN model.

As we can see from figure 12, the L1-Loss on the training samples for both the models has plateaued after a few epochs and converged to a relatively low value, which signifies that our QNN models are able to learn the internal distribution of the data, but, more interestingly, the loss of the QFT-based QNN model converged to a lower value compared to the standard hybrid QNN model, which compares well with our hypothesis. Although the QFT-based model demonstrates a better training convergence, the QFT block may be substituted by Classical Fast Fourier Transform since it provides no quantum speedup due to the performance bottleneck introduced by the preparation and readout of the states. We have included the QFT algorithm under the perspective of changing the classical node (convolutional and fully connected) by a quantum counterpart that does not need to read out classically the solution first.

5. Conclusion

This paper proposes a detailed methodology for the application of various quantum machine learning models for the prediction of molecular atomization energies. Firstly, we are able to efficiently encode each molecule into a vector using the information from the Coulomb matrix, atomic charges, and the Cartesian coordinates of the atoms in a respective molecule. Then we have successfully derived compressed representations of the molecules by training a deep convolutional autoencoder on the encoded vectors of the molecules. The reconstruction outputs of the autoencoder are in close agreement to the input encoded vectors. We have then used the feature vectors from the fully-connected bottleneck layer of the autoencoder as the compressed representations of the molecules and formed a dataset for the application quantum regression models. Before applying the quantum regression models, we have tested some classical deep learning and machine learning models to check the correlation between the feature vectors and the atomization energy of a particular molecule. We have calculated metrics such as MAE, RMSE, coefficient of determination (R2), and the Pearson correlation, and observed very good results indicating a close correlation between the feature vectors and the atomization energies of the molecules.

We have applied quantum machine learning models on the feature vectors for solving the regression problem of estimating the atomization energy of a molecule. We have studied four different quantum machine learning models, namely quantum linear regression, radial basis function neural network, a hybrid quantum neural network, and a QFT-based hybrid quantum neural network. The results of the quantum linear regression model are similar to the results obtained from a classical linear regression model. Also, we have proposed a new methodology for the implementation of a quantized version of the RBFNN model based on the trace distance of the density matrices of amplitude encoded states and the concept of Hilbert–Schmidt norm. We have successfully trained the Q-RBFNN model and presented the results which closely resembled those obtained from the classical RBFNN model. Lastly, we have implemented and studied the ability of a hybrid quantum neural network and a QFT-based hybrid QNN model to learn the internal representation of the data. We have studied the training convergence of both the models and presented two significant observations. Firstly, we are able to verify that the L1-loss metric converged to a low value for both the models, signifying that the models are able to map the feature vectors to the labels and secondly, we have seen that the loss corresponding to the training convergence of QFT-based hybrid QNN model converged to a lower value than that of the standard hybrid QNN model indicating that the Fourier basis states offer a better representation of the data compared to the standard basis states. The hybrid quantum neural network models do not seem to provide any speedup over their classical counterparts. While the results of quantum machine learning models discussed in this paper might not be comparable to the results of the current state-of-the-art classical machine learning models, they still offer a reasonable estimate of the capabilities of quantum and hybrid quantum algorithms for regression tasks, and the current NISQ systems might be suitable platforms for designing and developing quantum regression models. We hope to improve on the methods discussed in this paper and explore more quantum machine learning and QFT-based quantum machine learning methodologies. We hope that this paper becomes a precursor to more research in the field of quantum machine learning and motivate others to work on novel quantum machine learning methodologies for regression tasks in computational drug design.

Data availability statement

The data analyzed during this study are described in the following record: https://doi.org/10.1021/ja902302h and is openly available at http://quantum-machine.org/datasets/. The code that supports this study will be made open-source in the form of a GitHub repository within the first month from the date of publication.

ORCID iDs

Pranath Reddy https://orcid.org/0000-0002-6609-3495
Aranya B Bhattacherjee https://orcid.org/0000-0002-7713-8927

References

[1] Deo R C 2015 engMachine learning in medicine *Circulation* **132** 1920

[2] Sidey-Gibbons J A M and Sidey-Gibbons C J 2019 Machine learning in medicine: a practical introduction *BMC Med. Res. Methodol.* **19** 64

[3] Vamathevan J *et al* 2019 Applications of machine learning in drug discovery and development, nature reviews *Drug Discovery* **18** 463

[4] Rodrigues J, Jose F, Florea L, de Oliveira M C F, Diamond D, Oliveira J and Osvaldo N 2019 A survey on big data and machine learning for chemistry (arXiv:1904.10370 [physics.chem-ph])

[5] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press) (available at: http://www.deeplearningbook.org)

[6] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436

[7] Wang H and Raj B 2017 On the origin of deep learning (arXiv:1702.07800 [cs.LG])

[8] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information* 10th edn (Cambridge: Cambridge University Press)

[9] Hansen K, Montavon G, Biegler F, Fazli S, Rupp M, Scheffler M, von Lilienfeld O A, Tkatchenko A and Müller K-R 2013 Assessment and validation of machine learning methods for predicting molecular atomization energies *J. Chem. Theory Comput.* **9** 3404

[10] Tchagang A and Valdés J 2019 Prediction of the atomization energy of molecules using Coulomb matrix and atomic composition in a Bayesian regularized neural networks (arXiv:1904.10321 [physics.chem-ph])

[11] Wu Z, Ramsundar B, Feinberg E N, Gomes J, Geniesse C, Pappu A S, Leswing K and Pande V 2017 MoleculeNet: a benchmark for molecular machine learning (arXiv:1703.00564 [cs.LG])

[12] Blum L C and Reymond J-L 2009 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13 *J. Am. Chem. Soc.* **131** 8732

[13] Rupp M, Tkatchenko A, Müller K-R and von Lilienfeld O A 2012 Fast and accurate modeling of molecular atomization energies with machine learning *Phys. Rev. Lett.* **108** 058301

[14] Montavon G, Hansen K, Fazli S, Rupp M, Biegler F, Ziehe A, Tkatchenko A, Lilienfeld A V and Müller K-R 2012 Learning invariant representations of molecules for atomization energy prediction *Advances in Neural Information Processing Systems 25*, eds F Pereira, C J C Burges, L Bottou and K Q Weinberger (Red Hook, NY: Curran Associates, Inc.) pp 440–8 (available at: http://papers.nips.cc/paper/4830-learning-invariant-representations-of-molecules-for-atomization-energy-prediction.pdf)

[15] Kerenidis I, Landman J, Luongo A and Prakash A 2018 q-means: a quantum algorithm for unsupervised machine learning (arXiv:1812.03584 [quant-ph])

[16] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195

[17] Wan K H, Dahlsten O, Kristjánsson H, Gardner R and Kim M S 2017 Quantum generalisation of feedforward neural networks *npj Quantum Inf.* **3** 36

[18] Schuld M, Sinayskiy I and Petruccione F 2015 An introduction to quantum machine learning *Contemp. Phys.* **56** 172

[19] Cai X D, Wu D, Su Z E, Chen M C, Wang X L, Li L, Liu N L, Lu C Y and Pan J W 2015 Entanglement-based machine learning on a quantum computer *Phys. Rev. Lett.* **114** 110504

[20] Schuld M and Killoran N 2018 Quantum machine learning in feature Hilbert spaces (arXiv:1803.07128 [quant-ph])

[21] Bishwas A K, Mani A and Palade V 2020 An investigation on support vector clustering for big data in quantum paradigm *Quantum Inf. Process.* **19** 108

[22] Arodz T and Saeedi S 2019 Quantum sparse support vector machines (arXiv:1902.01879 [cs.LG])

[23] Anguita D, Ridella S, Rivieccio F and Zunino R 2003 Quantum optimization for training support vector machines *Neural Netw.* **16** 763 note advances in Neural Networks Research: IJCNN '03 (https://doi.org/10.1016/S0893-6080(03)00087-X)

[24] Beer K, Bondarenko D, Farrelly T, Osborne T J, Salzmann R, Scheiermann D and Wolf R 2020 Training deep quantum neural networks *Nat. Commun.* **11** 808

[25] Ricks B and Ventura D 2004 Training a quantum neural network *Adv. Neural Inf. Process. Sys.* **16** 1019–26

[26] Ezhov A A and Ventura D 2000 Quantum neural networks *Future Directions for Intelligent Systems and Information Sciences: The Future of Speech and Image Technologies, Brain Computers, WWW and Bioinformatics*, ed N Kasabov (Heidelberg: Physica-Verlag HD) pp 213–35

[27] Cong I, Choi S and Lukin M D 2019 Quantum convolutional neural networks *Nat. Phys.* **15** 1273

[28] Chakrabarti S, Huang Y, Li T, Feizi S and Wu X 2019 Quantum Wasserstein generative adversarial networks (arXiv:1911.00111 [quant-ph])

[29] Newman M E J 2010 *The Mathematics of Networks* (Oxford: Oxford University Press)

[30] Masci J, Meier U, Cireşan D and Schmidhuber J 2011 Stacked convolutional auto-encoders for hierarchical feature extraction *Artificial Neural Networks and Machine Learning—ICANN 2011* ed T Honkela, W Duch, M Girolami and S Kaski (Berlin, Heidelberg: Springer) pp 52–9

[31] Cheng Z, Sun H, Takeuchi M and Katto J 2018 Deep convolutional autoencoder-based lossy image compression (arXiv:1804.09535 [cs.CV])

[32] Chen M, Shi X, Zhang Y, Wu D and Guizani M 2017 Deep features learning for medical image analysis with convolutional autoencoder neural network *IEEE Trans. Big Data* **3** 1

[33] Rumelhart D E and McClelland J L 1987 Learning internal representations by error propagation *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations (Mitp)* (Cambridge, MA: MIT Press) pp 318–62 (available at: https://ieeexplore.ieee.org/document/6302929)

[34] Ruder S 2016 An overview of gradient descent optimization algorithms (arXiv:1609.04747)

[35] Ruder S 2016 An overview of gradient descent optimization algorithms (arXiv:1609.04747 [cs.LG])

[36] Joyce J M 2011 Kullback-leibler divergence *International Encyclopedia of Statistical Science*, ed M Lovric (Berlin, Heidelberg: Springer) pp 720–2

[37] Chen T and Guestrin C 2016 Xgboost: a scalable tree boosting system (arXiv:1603.02754 [cs.LG])

[38] Murtagh F 1991 Multilayer perceptrons for classification and regression *Neurocomputing* **2** 183

[39] Lecun Y and Bengio Y 1995 English (US)Convolutional networks for images, speech and time-series *English (US) The Handbook of Brain Theory and Neural Networks*, ed M Arbib (Cambridge, MA: MIT Press)

[40] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735

[41] Schuster M and Paliwal K 1997 Bidirectional recurrent neural networks, Trans *Sig. Proc.* **45** 2673–81

[42] Sak H, Senior A and Beaufays F 2014 Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition (arXiv:1402.1128 [cs.NE])

[43] Qiu J, Wang B and Zhou C 2020 Forecasting stock prices with long-short term memory neural network based on attention mechanism *PLoS One* **15** 1

[44] Elsworth S and Güttel S 2020 Time series forecasting using LSTM networks: a symbolic approach (arXiv:2003.05672 [cs.LG])

[45] Karim F, Majumdar S and Darabi H 2019 Insights into LSTM fully convolutional networks for time series classification *IEEE Access* **7** 67718

[46] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980 [cs.LG])

[47] Dean J *et al* 2012 Large scale distributed deep networks *Advances in Neural Information Processing Systems 25*, eds F Pereira, C J C Burges, L Bottou and K Q Weinberger (Red Hook, NY: Curran Associates, Inc.) pp 1223–31 (available at: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf)

[48] Chakraborty S, Gilyén A and Jeffery S 2018 The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation (arXiv:1804.01973 [quant-ph])

[49] Tang E 2018a A quantum-inspired classical algorithm for recommendation systems (arXiv:1807.04271 [cs.IR])

[50] Tang E 2018b Quantum-inspired classical algorithms for principal component analysis and supervised clustering (arXiv:1811.00414 [cs.DS])

[51] Wang G 2014 Quantum algorithm for linear regression (arXiv:1402.0660 [quant-ph])

[52] Schuld M, Sinayskiy I and Petruccione F 2016 Prediction by linear regression on a quantum computer *Phys. Rev.* A **94** 022342

[53] Johansson J R, Nation P D and Nori F 2013 QuTiP 2: a Python framework for the dynamics of open quantum systems *Comput. Phys. Commun.* **184** 1234

[54] Shao C 2020 Data classification by quantum radial-basis-function networks *Phys. Rev.* A **102** 042418

[55] Giovannetti V, Lloyd S and Maccone L 2008 Quantum random access memory *Phys. Rev. Lett.* **100** 160501 (arXiv:0708.1879 [quant-ph])

[56] Lloyd S, Mohseni M and Rebentrost P 2013 Quantum algorithms for supervised and unsupervised machine learning (arXiv:1307.0411 [quant-ph])

[57] Crooks G E 2019 Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition (arXiv:1905.13311 [quant-ph])