Evaluating the Performance of Some Local Optimizers for Variational Quantum Classifiers

Nisheeth Joshi\textsuperscript{1}, Pragya Katyayan\textsuperscript{1}, Syed Afroz Ahmed\textsuperscript{2}

\textsuperscript{1}Department of Computer Science, Banasthali Vidyapith, India
\textsuperscript{2}Umm Al Quwain University, United Arab Emirates

Email: nisheeth.joshi@rediffmail.com

Abstract. In this paper, we have studied the performance and role of local optimizers in quantum variational circuits. We studied the performance of the two most popular optimizers and compared their results with some popular classical machine learning algorithms. The classical algorithms we used in our study are support vector machine (SVM), gradient boosting (GB) and random forest (RF). These were compared with a variational quantum classifier (VQC) using two sets of local optimizers viz AQGD and COBYLA. For experimenting with VQC, IBM Quantum Experience and IBM Qiskit was used while for classical machine learning models, scikit-learn was used. The results show that machine learning on noisy immediate scale quantum machines can produce comparable results as on classical machines. For our experiments, we have used a popular restaurant sentiment analysis dataset. The extracted features from this dataset and then after applying PCA reduced the feature set into 5 features. Quantum ML models were trained using 100 epochs and 150 epochs on using EfficientSU2 variational circuit. Overall, four Quantum ML models were trained and three Classical ML models were trained. The performance of the trained models was evaluated using standard evaluation measures viz, Accuracy, Precision, Recall, F-Score. In all the cases AQGD optimizer-based model with 100 Epochs performed better than all other models. It produced an accuracy of 77\% and an F-Score of 0.785 which were highest across all the trained models.

1. Introduction

Our understanding of machine learning (ML) (and deep learning (DL) concepts has come of age. Mostly everything that can be studied has already been studied and now we are moving towards optimizing the performance of the ML/DL models. This quest for continuous advancement has made us look into quantum mechanics where we try to couple the concepts of quantum physics into machine learning. The synergy of these two concepts had given rise to a new area of study termed quantum machine learning (QML). Until now ML/DL algorithms could process a high volume of data and based on it could make intelligent decisions. QML has made the process even better as now we can learn new patterns from data that were previously termed as computational impossible. This is achieved by converting the classical input data into quantum states, thus making the process easier for a quantum computer to extract patterns of interest from them.

In quantum settings, algorithms can be of two types. There are classical algorithms and quantum algorithms. Classical algorithms are ones that use mostly classical constructs as used in classical
machines. Quantum algorithms work with quantum states to make predictions. Similarly, data can be of two types. It can be classical data or quantum data. Thus, four different types of machine learning models can be developed using this information. These are ML models with classical algorithms working on classical data. ML modes with classical algorithms with quantum data. ML models with quantum algorithms with classical data and ML models with quantum algorithms with quantum data. Figure 1 shows these four types of models with two sets of algorithms (classical and quantum) and two types of data (classical and quantum).

Figure 1: Four different approaches to combine the disciplines of quantum computing and machine learning [1]

The rest of the paper is organized as: section 2 shows the work done in the area of quantum machine learning. Section 3 shows our experiments, the design of a quantum classification algorithm and how we tuned it specifically to our dataset. Section 4 shows the results of classification from quantum machine learning algorithms and classical machine learning algorithms and how good or bad did the quantum algorithm performed in comparison to its classical counterparts. Finally, Section 5 concludes the paper.

2. Literature Review
Since we are quantum computers are still under development, so is quantum machine learning. At the current state of the art, we can perform are experiments either on quantum simulators or noisy immediate scale quantum (NISQ) computers which are still being researched. We still have a few years where we can have a fully functional noise-tolerant quantum computer, but this should not stop us from the joy of exploring new possibilities. As quantum computing is advancing each day so does quantum machine learning.

Kak [2] showed how a quantum neural network can be used and how it would more versatile than classical artificial neural networks. Menneer and Narayanan [3] showed how multiple single-layered neural networks can be used to form a much larger quantum neural network. Perus [4] was the first to show the use of gradient descent with quantum gates (more specifically CNOT gates) and developed a mechanism to facilitate parallelism in quantum neural networks. Faber et al. [5] argued how artificial neural networks can be implemented on quantum hardware. Schuld et al. [6] showed how quantum neural networks can be used in classification by using a distance function. By doing so they showed how quantum phenomenon can be in line with quantum theory.

Weber at el. [7] showed how quantum machine learning (specifically quantum neural networks) can make use of classical computers to model classification problems. Mitarai et al. [8] proposed a hybrid algorithm for quantum machine learning which they termed quantum circuit learning and showed how it can be tuned by using parameters. They further showed how a high-depth circuit can be implemented.
They theoretically showed the performance of their circuit learning and also did mathematical simulations to prove this. Abhijit et al. [9] showed how programmers working on classical computers can be trained to program quantum computers. They developed a complete tutorial for this. Kopczyk [10] showed how data science can be used in quantum computing in general and quantum machine learning in particular for their analysis tasks. They made a detailed step by step explanation of all the algorithms that are being used in quantum machine learning.

Schuld et al. [11] did an evaluation of gradients on quantum computers. They showed how the variational algorithm performs when gradients are used for optimization. They showed procedure of estimating expectation values of gradients for quantum measurements. Fastovets et al. [12] proposed approaches through which classical machine learning algorithms can be executed on quantum computers. They demonstrated their approach by executing a multiclass tensor network algorithm on a quantum computer provided by IBM Quantum Experience. Further, they showed how quantum tomography problem can be approached through artificial neural networks. They showed how their approach can predict the quantum state by reducing the noise. Zhao et al. [13] showed how the problem of large measurement computation in variational algorithms can be reduced using constant factor speedup for lattice and random Pauli Hamiltonians and showed promising results.

Lu et al. [14] adversarial machine learning on quantum computers. They showed how quantum machine learning-based classifiers which produce good results can be deceived by adversarial examples that use imperceptible perturbations to the original legitimate samples. Lloyd et al. [15] proposed quantum metric learning where state preparation tasks in a quantum circuit learning can be minimized and would also help in better measurement calculation. Terashi et al. [16] did a comparison of classical machine learning algorithms with quantum machine learning algorithms in high-energy physics applications. Through experiments, they showed that quantum machine learning can be used for this task. They specifically studied variational quantum algorithms.

Wu et al. [17] showed how a robust machine learning algorithm can be used on a noisy immediate scale quantum computer. In their algorithm, they incorporate feature selection by encoding input data to quantum states. Zhang et al. [18] argued that quantum machine learning algorithms which show speed up over classical machine learning algorithm could not keep this advantage while decoding the quantum states into final classical output. They proposed a protocol to address this issue by using the Gram-Schmidt orthonormal procedure. Garg and Ramakrishnan [19] reviewed advances made in quantum deep learning. They showed how quantum deep neural networks have advanced and also showed how can be used in natural language processing tasks. Guan et al. [20] investigated if quantum machine learning approaches can be used in the area of high-energy physics. They gave a very detailed description of this by providing a review of the work done in this area. Guan et al. [21] developed an algorithm that could verify if a quantum machine learning algorithm is robust on a particular set of training data. Through experiments, they showed how this approach improves the accuracy of the quantum machine learning model.

Suzuki and Katouda [22] showed how parameterized quantum circuits can be used to predict toxicity in organic chemistry. They studied the toxicity of some phenol structures. Through experiments, they showed that their quantum machine learning approach performed significantly better than the existing linear regression-based methods. Blance and Spannowsky [23] studied the use of quantum machine learning in particle physics. They proposed an algorithm using a quantum variational circuit that used multiple gradient descent approaches which optimized their results. They applied their algorithm on resonance search in di-top final states and showed that their approach performs better than the approach is being used currently.

This is becoming promising as large corporations like Google [24] and IBM [25] are researching quantum computers and have also provided open-source libraries to play with their quantum hardware. More specifically with quantum machine learning and test the limits of their hardware. Recently Google has also released Tensorflow Quantum which is built on top of their popular Tensorflow libraries and can be used to play with quantum machine learning.
3. Methods and Materials

This section explains the experiment that was performed using Qiskit Toolkit which is developed for experimenting with noisy immediate scale quantum (NISQ) machines. We have applied sentiment analysis for our experimentation and compared the performance of quantum machine learning models with classical machine learning models. We have checked the performance of two local optimizers of quantum machine learning and have tried to identify which one gives better performance on our dataset.

3.1. Pre-processing

For training any machine learning model, the first step is pre-processing as it helps in the extraction and selection of features. We have used a restaurant review dataset which has positive and negative reviews. Since the data was textual, we extract features from these textual features. The first feature that we extracted was the number of punctuations that the dataset had. Next, we calculated the length of the review. Then we removed the stop words from the corpus and calculated the tf-idf frequencies for unigram lexicons in the corpus. This generated 1698 tf-idf frequencies (features). This was a matrix of 1000x1698 data points, which had 1000 rows and 1698 columns.

For training a machine learning model on a computer, we need to assign one feature to one qubit. Since the current state of the art quantum machines do not provide such a high number of qubits, we were needed to scale down the number of features so that they can be processed with quantum computers (Maximum features that can be supported are thirty-two). Thus we transformed this 1000x1698 data point matrix into a matrix of 1000x1 by multiplying a unitary matrix of 1000x1.

The following explanation shows the working of this process, where we have a matrix of m rows and n columns which needs to be transformed into a matrix of m rows and 1 column.

\[
\begin{bmatrix}
a_{11}a_{12}a_{13} \ldots & a_{1n} \\
a_{21}a_{22}a_{23} \ldots & a_{2n} \\
\vdots & \vdots \\
a_{m1}a_{m2}a_{m3} \ldots & a_{mn}
\end{bmatrix}_{m \times n} \times 
\begin{bmatrix}
1 \\
1 \\
\vdots \\
1
\end{bmatrix}_{n \times 1} = 
\begin{bmatrix}
a_{11} \times 1 + a_{12} \times 1 + a_{13} \times 1 + \ldots + a_{1n} \times 1 \\
a_{21} \times 1 + a_{22} \times 1 + a_{23} \times 1 + \ldots + a_{2n} \times 1 \\
\vdots \\
a_{m1} \times 1 + a_{m2} \times 1 + a_{m3} \times 1 + \ldots + a_{mn} \times 1
\end{bmatrix}_{m \times 1}
\]

Similarly, we computed the tf-idf frequencies for bigram and trigram word sequences. Other features that we extracted from the dataset were:

1. Number of Stopwords.
2. Number of Content Words
3. Number of Nouns
4. Number of Verbs
5. Number of words on which lemmatization was done
6. Number of words with all letters capital
7. Number of elongated words
8. Number of positive emoticons
9. Number of negative emoticons
10. Number of neutral emoticons
11. Number of slang words
12. Number of negation words
13. Total positive score of words in the negated context
14. Total negative score of words in the negated context
15. Total positive score of words in a positive context
16. Total negative score of words in a negative context

Finally, we had a total of 17 features in our feature set. Next, we performed feature selection using principal component analysis (PCA), which reduced the features to just five features. With these five features, we had trained our classical and quantum classifiers.

3.2. Structure of a Variational Quantum Classifier

Variational Quantum Classifiers (VQC) are artificial neural networks. They are very popular as one can easily train a classifier without performing error correction which is required while working on NISQ machines as these machines tend to add noise in the output. VQC is considered as a hybrid classifier where a part of the processing is done on a classical computer viz parameter optimization and updation, and the cost calculation is done at the quantum computer which finally helps in calculating the error and accuracy of the model.

Thus, to perform machine learning using VQC, we need to design a quantum circuit that can act as a machine learning classifier. This is achieved by designing a circuit which can have multiple parameters that are optimized to produce a minimum loss. The trained circuit can be shown using equation 2.

\[
f(w, b, x) = y
\]

Here, \( f \) is the classifier that we are training through VQC and \( y \) is the output label that we which to produce through our classifier. The objective of the function (classifier) \( f \) is to reduce the loss function. This is similar to the artificial neural networks that we train in classical machine learning. The classifier is a circuit-centric quantum classifier [26] which has three sub-parts viz (1) state preparation circuit, (2) model circuit and (3) measurement and post-processing.

The initial state or state preparation circuit takes the feature vector as an input with \( n \) features and encodes them into \( n \) qubits. The model circuit is the main unit which applies various quantum gates to this input state and tries to minimize the loss. Finally, we measure the output label in the third sub-circuit which performs measurement and post-processing on the output received. The entire working of this entire VQC is shown in figure 2.

In figure 2, the feature map and variational circuit are the elaborations of the model circuit of figure 1. In both the figures, our dataset is transformed into a feature vector \( \hat{f} \) which is then supplied for state preparation circuit which then converts the features in feature vector into qubits. These qubits are then sent to the model circuit where we qubits are first sent the quantum feature map which is a black box. The role of this box is to encode classical data \( x_i \) into quantum state \( |\varphi(x_i)\rangle \). This is done by transforming the ground state \( |0\rangle^\otimes n \) into products of single and unitary phase gates. Here,

\[
V_\varphi(x) = V_\varphi(x)H^\otimes n
\]

Here \( H \) represents a Hadamard gate. The final computation is done by:

\[
V_\varphi(x) = \exp\left(i\sum_{s \in |n|} \varphi_s(x) \prod_{x \subseteq S} z_x\right)
\]
Here $V_\emptyset(x)$ is a diagonal gate which assumes Pauli-Z. Next, the results of this process are supplied to the variational circuit $U(\theta)$ which had $l$ layers of $\theta$-parameters which are optimized during training by minimizing the cost function in the classical machine and thus tuning $\theta$ recursively. This parameterized approach to machine learning in a quantum computer is also referred to as quantum circuit learning [26] (QCL). QCL uses multiple exponential functions with the $n$ qubits from the parameterized circuit. This is something that is not possible on classical machines; thus, this provides the quantum advantage over classical computing, as here we can represent a larger set of complex functions than what the classical computers can handle.

3.3. Design of Variational Quantum Classifier for Sentiment Analysis using Custom (Parameterised) Circuits

In order to develop a classifier, we were first required to design a circuit that could learn the patterns. In order to do so, we first configured some parameters which were required to train the final circuit. These were:

3.3.1. Feature Map: A feature map is the fundamental building block in a machine learning setup. It is a function that maps an input feature vector to feature space. We have been using feature maps in pattern recognition and computer vision-based systems. Feature maps are used because they help the learning algorithm to perform better and predict accurately. This is so because feature maps help in dimensionality reduction as it reduces the resources required in defining the enormous data. Moreover, if larger data is used, then it might cause overfitting of the model. To some extent, feature maps keep a check on this.

Broadly they have been used in several learning algorithms, but have gained popularity with the advent of kernel methods where they have been extensively used. With the dawn of deep learning systems, there has been a renewed interest in the machine learning community in customizing a variety of feature maps.

Since VQC is a kind of a neural network, it also uses a feature map. Many feature maps are available in qiskit, but we have used Pauli feature maps for our experiments. Equation 4 shown above is the formulation of a Pauli feature map. The $Z_i$ in this equation denotes a Pauli matrix. In our experiment, we have used this feature map and have customized it with 5 input strings (one each for 5 qubits). Figure 3 shows the circuit diagram for these feature maps. We have used 3 repetitions of this feature map in our experiment which means the circuit in figure 3 is drawn (used) three times in our classifier.
3.3.2. Variational Circuit: A variational quantum circuit is a computational routine consisting of coherent quantum operations on quantum data, such as qubits, and concurrent real-time classical computation. It is an ordered sequence of quantum gates, measurements and resets, all of which may be conditioned on and use data from the real-time classical computation.

In our experiment, we have used the EfficientSU2 variational circuit. This circuit was used with a Pauli feature map with two different sets of epochs (100 and 150). Figure 4 shows the diagram for the EfficientSU2 variational circuit.

3.3.3. Optimizer and Measurement: In VQC the optimizer gets the results from the classifier and compares it with the actual result. Based on this it calculates the error (loss) in the training process and thus sends the feedback to the classifier which helps in tuning the weights (feature-map parameters) which can retain the variational circuit. This process is repeated until the loss is minimized. Finally, the optimizer results in the list of parameters that have the minimum error (loss) in training. In our experiment, we have used two different optimizers viz. COBYLA [27] and AQGD [28]. COBYLA (Constrained Optimization By Linear Approximation) optimizer is a numerical optimization method.
used to minimize the loss in constraint problems where the derivative of the objective is not known. Here the optimization is performed by linear approximation which is applied on constraints and the objective function. On the other hand, AQGD (Analytic Quantum Gradient Descent) optimizer is a quantum variant of the popular Gradient Descent function used in artificial neural networks.

Thus, we have trained four quantum classifiers, with each epoch of our variational circuit, we have used two optimizers i.e. Efficient SU2 with 100 Epochs and COBYLA optimizer, EfficientSU2 with 150 epochs and COBYLA optimizer, Efficient SU2 with 100 Epochs and AQGD optimizer, and finally, EfficientSU2 with 150 epochs and AQGD optimizer,

![Diagram of EfficientSU2 Variational Circuit for Sentiment Analysis Task](image)

3.4 Classical Machine Learning Algorithms: In order to compare the results of our quantum machine learning classifier, we also trained three popular classical machine learning models viz support vector machines, random forest and gradient boosting. Among these gradient boosting and the random forest are considered as ensemble classifiers. Support vector machines on the other hand is a popular machine learning algorithm that has shown promising results in several machine learning tasks.

4. Results and Discussion

We had a corpus of 1000 data points with 500 being positive reviews and 500 being negative reviews. Among these, we divided the corpus into 3 sets viz training, validation and the test set. The training set and the validation set were used at the time of training the models. The models were trained on a training set and were optimized using the validation set. Once the models were trained, we used the test set which was completely unseen by the models. This gave fair chances to all the models and hence negated the possibility of biasness of data towards any particular model. The statistics of the three sets are shown in table 1.

| Set        | Data points |
|------------|-------------|
| Training Set | 600         |
| Validation Set | 200       |
| Test Set    | 200         |

Table 1: Statistics of Training, Validation and Test Sets

Once the models were tested, we generated the confusion matrix of each of the models where we calculated the true positives, true negatives, false positives and false negatives. Table 2 shows a general confusion matrix where we can see how the results of the model are compared with the actual predictions. Here, TP is the number of true positives i.e. the positive tuples which were correctly identified by the model. TN is the number of true negatives i.e. the negative tuples which were correctly
identified. FP is the number of false positives i.e. the negative tuples which were incorrectly identified as the positive tuples and FN is the number of false negatives i.e. the positive tuples which were incorrectly identified as negative. P is the total actual positive tuples and N is the total actual negative tuples. P’ is the total positive tuples that the model has predicted and N’ is the total negative tuples that the model has predicted.

| Actual | Predicted | Total |
|--------|-----------|-------|
| Positive | TP | FN | P |
| Negative | FP | TN | N |
| Total | P' | N' | P+N |

Table 2: General Confusion Matrix

The confusion matrix of Gradient Boosting is shown in table 3. Here we had 64 true positives, 45 false negatives, 26 false positives and 65 true negatives. The confusion matrix of Random Forest is shown in table 4. Here we had 67 true positives, 42 false negatives, 23 false positives and 68 true negatives. Table 5 shows the confusion matrix of the Support Vector Machines Model. Here we had 81 true positives, 28 false negatives, 29 false positives and 62 true negatives. Table 6 shows the confusion matrix of the EfficientSU2 Model with 100 epochs and COBYLA optimizer. Here we had 81 true positives, 28 false negatives, 23 false positives and 68 true negatives.

Table 3: Confusion Matrix of Gradient Boosting Model

Table 4: Confusion Matrix of Random Forest Model

Table 5: Confusion Matrix of Support Vector Machines Model

Table 6: Confusion Matrix of EfficientSU2 Model with 100 Epochs and COBYLA Optimizer
Table 7 shows the confusion matrix of the EfficientSU2 Model with 150 epochs and COBYLA. Here we had 74 true positives, 35 false negatives, 21 false positives and 70 true negatives. Table 8 shows the confusion matrix of the EfficientSU2 Model with 100 epochs with AQGD Optimizer. Here we had 84 true positives, 25 false negatives, 21 false positives and 70 true negatives.

| Actual | Predicted | Positive | Negative | Total |
|--------|-----------|----------|----------|-------|
| Positive | 74        | 35       | 109      |
| Negative | 21        | 70       | 91       |
| Total   | 95        | 105      | 200      |

Table 7: Confusion Matrix of EfficientSU2 Model with 150 Epochs and COBYLA Optimizer

| Actual | Predicted | Positive | Negative | Total |
|--------|-----------|----------|----------|-------|
| Positive | 84        | 25       | 109      |
| Negative | 21        | 70       | 91       |
| Total   | 104       | 96       | 200      |

Table 8: Confusion Matrix of EfficientSU2 Model with 100 Epochs and AQGD Optimizer

Table 9 shows the confusion matrix of the EfficientSU2 Model with 150 epochs and AQGD Optimizer. Here we had 79 true positives, 30 false negatives, 23 false positives and 68 true negatives.

| Actual | Predicted | Positive | Negative | Total |
|--------|-----------|----------|----------|-------|
| Positive | 79        | 30       | 109      |
| Negative | 23        | 68       | 91       |
| Total   | 101       | 99       | 200      |

Table 9: Confusion Matrix of EfficientSU2 Model with 150 Epochs

The results of the seven models are shown in table 10. We evaluated the performance of the models based on 5 evaluation parameters. These were accuracy, specificity, precision, recall and f-score. The accuracy of the model over a test set is tuples that are correctly classified by the model. It calculated using equation 5. Precision and Recall are the most popular measures which are used in the evaluation of classification models. Precision is the percentage of tuples labeled as positive, which are positive. It is also known as the measure of exactness and is calculated using equation 6. While Recall is the percentage of positive tuples that are labeled as such. It is also known as the measure of completeness and is calculated using equation 7. F-score is a combination of precision and recall. This combines them into a single measure so that we may calculate exactness and completeness at the same time. This is calculated using equation 8.

\[ Accuracy = \frac{TP + TN}{P + N} \]  

\[ Precision = \frac{TP}{TP + FP} \]  

\[ Recall = \frac{TP}{TP + FN} \]  

\[ F-score = \frac{2 \times Precision \times Recall}{Precision + Recall} \]  

\[ TP = \text{True Positives} \]  

\[ TN = \text{True Negatives} \]  

\[ FP = \text{False Positives} \]  

\[ FN = \text{False Negatives} \]  

\[ P = \text{Positive} \]  

\[ N = \text{Negative} \]
\[ Precision = \frac{TP}{TP+FP} \] 

(6)

\[ Recall = \frac{TP}{TP+FN} \] 

(7)

\[ F - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \] 

(8)

| Model                                      | Accuracy | Precision | Recall  | F-Score  |
|---------------------------------------------|----------|-----------|---------|----------|
| Gradient Boosting                           | 0.645    | 0.7111    | 0.5872  | 0.6432   |
| Random Forest                              | 0.675    | 0.7444    | 0.6147  | 0.6734   |
| Support Vector Machine                      | 0.715    | 0.7364    | 0.7431  | 0.7397   |
| EfficientSU2 100 Epochs with COBYLA        | 0.72     | 0.7789    | 0.6789  | 0.7255   |
| EfficientSU2 150 Epochs with COBYLA        | 0.77     | 0.8       | 0.7706  | 0.785    |
| EfficientSU2 100 Epochs with AQGD          | 0.77     | 0.8       | 0.7706  | 0.785    |
| EfficientSU2 150 Epochs with AQGD          | 0.735    | 0.7745    | 0.7248  | 0.7488   |

Table 10: Evaluation results of Classical and Quantum Machine Learning Models

From the above table, we can see that EfficientSU2 with 100 epochs and AQGD Optimizer produced better results than the other models. The accuracy of this model was 77% which was the highest among all other models. This model also had a better precision, recall, and f-score with 80%, 77.06%, and 78.5% respectively.

5. Conclusion and Future Work

In this paper, we tried analyzing the performance of quantum machine learning models with classical machine learning models. We found that quantum ML models perform slightly better than classical machine learning models. We had used three classical machine learning models and compared their performance with four quantum machine learning models. We used four quantum machine learning models viz EfficientSU2 with 100 and 150 epochs with COBYLA and AQGD optimizers. Through experiments, we found that the AQGD optimizer with 100 epochs performed better than all the other models.

As an extension to this study, we would like to perform an experiment with other parameters and further improve the performance of our quantum machine learning model. We would like to experiment with different optimizers and feature maps and see the change in performance. Further, we would also like to experiment with datasets in different domains so that we may establish the true quantum
advantage of QML models over classical ML models. When will have the actual noiseless quantum computers, we would also like to perform these experiments on them as well.

Acknowledgment
We would like to thank the IBM Quantum team for providing the access to the quantum computers through IBM Quantum Experience and also the Qiskit library which helped us in performing these experiments.

References
[1] Aïmeur, E., Brassard, G., & Gambs, S. (2006, June). Machine learning in a quantum world. In Conference of the Canadian Society for Computational Studies of Intelligence (pp. 431-442). Springer, Berlin, Heidelberg.
[2] S. C. Kak, Quantum neural computing, Adv. Imag. Elect. Phys. 94, 259 (1995).
[3] T. Mennear and A. Narayanan, Quantum-inspired neural networks, Tech. Rep. R329, (1995).
[4] M. Perus. Neuro-quantum parallelism in brain-mind and computers, Informatica 20, 173 (1996).
[5] J. Faber and G. A. Giraldi, Quantum Models for Artificial Neural Networks, LNCC.
[6] M. Schuld, I. Sinayskiy, and F. Petruccione, The quest for a quantum neural network, Quantum Inf. Process. 13, 2567 (2014).
[7] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum deep learning, arXiv:1412.3489, (2014).
[8] Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. Physical Review A, 98(3), 032309.
[9] Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, Andreas Bärtschi, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O’malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Smitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Patrick J. Coles, Marc Vuffray, And Andrej. Lokhov (2018). Quantum algorithm implementations for beginners. arXiv, arXiv-1804.
[10] Coles, P. J., Eidenbenz, S., Pakin, S., Adedoyin, A., Ambrosiano, J., Anisimov, P., ... & Gunter, D. (2018). Quantum algorithm implementations for beginners. arXiv, arXiv-1804.
[11] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. Physical Review A, 99(3), 032331.
[12] Fastovets, D. V., Bogdanov, Y. I., Bantysh, B. I., & Lukichev, V. F. (2019, March). Machine learning methods in quantum computing theory. In International Conference on Micro-and Nano-Electronics 2018 (Vol. 11022, p. 110222S). International Society for Optics and Photonics.
[13] Zhao, A., Tranter, A., Kirby, W. M., Ung, S. F., Miyake, A., & Love, P. J. (2020). Measurement reduction in variational quantum algorithms. Physical Review A, 101(6), 062322.
[14] Lu, S., Duan, L. M., & Deng, D. L. (2020). Quantum adversarial machine learning. Physical Review Research, 2(3), 033212.
[15] Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., & Killoran, N. (2020). Quantum embeddings for machine learning. arXiv preprint arXiv:2001.03622.
[16] Terashi, K., Kaneda, M., Kishimoto, T., Saito, M., Sawada, R., & Tanaka, J. (2020). Event Classification with Quantum Machine Learning in High-Energy Physics. arXiv preprint arXiv:2002.09935.
[17] Wu, R. B., Cao, X., Xie, P., & Liu, Y. X. (2020). End-to-End Quantum Machine Learning with Quantum Control Systems. arXiv preprint arXiv:2003.13658.
[18] Zhang, K., Hsieh, M. H., Liu, L., & Tao, D. (2020). Efficient State Read-out for Quantum Machine Learning Algorithms. arXiv preprint arXiv:2004.06421.
[19] Garg, S., & Ramakrishnan, G. (2020). Advances in Quantum Deep Learning: An Overview. arXiv preprint arXiv:2005.04316.
[20] Guan, W., Perdue, G., Pesah, A., Schuld, M., Terashi, K., Vallecorsa, S., & Vlimant, J. R. (2020). Quantum Machine Learning in High Energy Physics. arXiv preprint arXiv:2005.08582.
[21] Guan, J., Fang, W., & Ying, M. (2020). Robustness verification of quantum machine learning. arXiv preprint arXiv:2008.07230.
[22] Suzuki, T., & Katouda, M. (2020). Predicting toxicity by quantum machine learning. arXiv preprint arXiv:2008.07715.
[23] Blance, A., & Spannowsky, M. (2020). Quantum Machine Learning for Particle Physics using a Variational Quantum Classifier. arXiv preprint arXiv:2010.07335.
[24] Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Philip M., Murphy Y. N., Ramin H., Evan P., Martin L., Andrea S., Michael S., David V. D., Jarrod R. M., Sergio B., Dave B., Alan K. H., Hartmut N., Masoud M. (2020). Tensorflow quantum: A software framework for quantum machine learning. arXiv preprint arXiv:2003.02989.
[25] Gadi Aleksandrowicz; Thomas Alexander; Panagiotis Barkoutsos; Luciano Bello; Yael Ben-Haim; David Bucher; Francisco Jose Cabrera-Hernández; Jorge Carballo-Franquis; Adrian Chen; Chun-Fu Chen; Jerry M. Chow; Antonio D. Córcoles-Gonzales; Abigail J. Cross; Andrew Cross; Juan Cruz-Benito; Chris Culver; Salvador De La Puente González; Enrique De La Torre; Delton Ding; Eugene Dumitrescu; Ivan Duran; Pieter Eendebak; Mark Everitt; Ismael Faro Sertage; Albert Frisch; Andreas Fuhrer; Jay Gambetta; Borja Godoy Gago; Juan Gomez-Mosquera; Donny Greenberg; Ikko Hamamura; Vojtech Havlícek; Joe Hellmers; Lukasz Herok; Hiroshi Horii; Shaoan Hu; Takashi Imamichi; Toshinari Itoko; Ali Javadi-Abhari; Naoki Kanazawa; Anton Karazeev; Kevin Krsulich; Peng Liu; Yang Luh; Yunho Maeng; Manoel Marques; Francisco Jose Martin-Fernández; Douglas T. McClure; David McKay; Srujan Meesala; Antonio Mezzacapo; Nikolaj Moll; Diego Moreda Rodriguez; Giacomo Nannicini; Paul Nation; Pauline Ollitrault; Lee James O'Riordan; Hanhee Paik; Jesús Pérez; Anna Phan; Marco Pistoi; Viktor Prutyanov; Max Reuter; Julia Rice; Abdón Rodriguez Davila; Raymond Harry Putra Rudy; Mingi Ryu; Ninad Sathaye; Chris Schnabel; Eddie Schoute; Kanav Setia; Yunong Sh; Adenilton Silva; Yukio Siraichi; Seyon Sivaranah; John A. Smolin; Mathias Soeken; Hitomi Takahashi; Ivano Tavernelli; Charles Taylor; Pete Taylour; Kenso Trabing; Matthew Treinish; Wes Turner; Desiree Vogt-Lee; Christophe Vuillot; Jonathan A. Wildstrom; Jessica Wilson; Erick Winston; Christopher Wood; Stephen Wood; Stefan Wörner; Ismail Yunus Akhalwaya; Christa Zoufal (2019). Qiskit: An open-source framework for quantum computing. available at https://zenodo.org/record/2562111#.X5pX9NQzbiU
[26] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. Phys. Rev. A, 98:032309, Sep 2018.
[27] Powell, M J D. A direct search optimization method that models the objective and constraint functions by linear interpolation. 1994. Advances in Optimization and Numerical Analysis, eds. S. Gomez and J-P Hennart, Kluwer Academic (Dordrecht), 51-67.
[28] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. Physical Review A, 99(3), 032331.