Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 568–575

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Training an Artificial Neural Network Using Qubits as Artificial Neurons: A Quantum Computing Approach

Avinash Chalumuri[a], Raghavendra Kune[b], B. S. Manoj[a,*]

[a] *Indian Institute of Space Science and Technology, Thiruvananthapuram 695547, India*
[b] *Advanced Data Processing Research Institute, Hyderabad 500009, India*

Abstract

Artificial neural networks (ANN) proved to be efficient in solving many problems for big data analytics using machine learning. The complex and non-linear features of the input data can be learned and generalized by ANN. In the big data era, enormous amounts of data arrive from multiple sources. A stage is expected to be reached where even supercomputers are likely inundated with the big data. Training an ANN in such a situation is a challenging task due to the size and dimension of the big data. Also, a large number of parameters are to be used and optimized in the network to learn the patterns and analyze such data. Quantum computing is emerging as a field that provides a solution to this problem as a quantum computer can represent data differently using qubits. Qubits on quantum computers can be used to detect the hidden patterns in data that are difficult for a classical computer to find. Hence, there exists a huge scope for application in the area of artificial neural networks. In this work, we primarily focused on training an artificial neural network using qubits as artificial neurons. The simulation results show that our quantum computing approach for ANN (QC ANN) is efficient when compared to classical ANN. The model with qubits as artificial neurons can learn the features of data using fewer parameters for a binary classification task. We demonstrate our experiment using a quantum simulator and optimization of the quantum parameters used in QC ANN is carried out on a classical computer.

Keywords:
Artificial neural networks; Quantum computing; Binary classification;

1. Introduction

Data processing is an area of concern, due to the rapid speeds at which the volume of data is growing. Getting some insights from very large amounts of data is a big challenge. IDC (International Data Corporation) in its white paper Data Age 2025 [15] predicts that, worldwide data will grow from 33ZB (1ZB = 10^{21}bytes) in 2018 to 175ZB

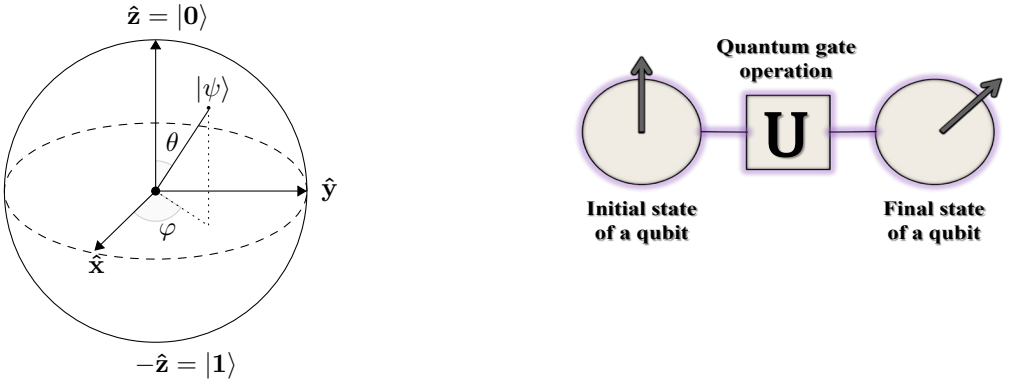
* Corresponding author. Tel.: +91-940-001-6607
E-mail address: bsmanoj@iist.ac.in

10.1016/j.procs.2020.04.061

by 2025. Artificial neural networks [16, 12, 24] removed a lot of obstacles in big data processing, thereby, turning the data to information for successful decision making. Data growth and size create new problems to be addressed and challenge the existing computing power of artificial neural networks. An ANN can learn complex relationships from the input data using the parameters between the layers of the network. These parameters are optimized using backpropagation during the training phase. As the size and dimension of the input data grow, training a neural network with fewer parameters is a difficult task. Complex machine learning problems can be solved efficiently using a quantum computer [5]. Biamonte et.al [3] described the advantages of using quantum computing in solving machine learning problems. In their work [18, 19], Schuld et.al designed a quantum circuit as a classifier for binary classification using distance-based kernel function. Recent work [8, 10] states that machine learning problems can be solved efficiently using a hybrid classical-quantum approach that uses a classical computer to optimize quantum parameters. Tacchino et.al [22] implemented an artificial neuron on a quantum processor. Open-source software [6] are developed to implement quantum algorithms on a real quantum processor or a quantum simulator. Classical computing can store and process the data in the form of binary digits. In contrast, quantum computers store and process the data using qubits [13] which use both 0 and 1 at the same time. This counter-intuitive quantum-mechanical property is called superposition. Entanglement is another quantum-mechanical property that is used in many algorithms to create interactions between qubits. The main aim of this work is to find the advantage of training a neural network using a near-term quantum computer with fewer parameters.

Solving machine learning problems using the present state of a quantum computer has its own limitations. Quantum computers are complex systems due to their need for perfect isolation from the environment. Lack of isolation results in the qubits of quantum computers losing coherence. This property of decoherence of qubits is a challenging problem when designing a quantum algorithm as a quantum circuit on a near-term quantum computers [14].

We introduce quantum computing in Section 2 and our approach for training an ANN using quantum computing in comparison with classical ANN is given in Section 3. Details of our experimental approach with results are provided in Section 4. Finally, we conclude the paper in Section 5 with the future scope of our work.



(a) A qubit state $|\psi\rangle$ respresented on a bloch sphere

(b) Gate operation on a qubit

Fig. 1. (a) The state of a qubit is represented on a bloch sphere as $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)\exp(i\phi)|1\rangle$; (b) Quantum gate operation is performed as a unitary transformation denoted by U, that transforms a qubit form one state to another state.

2. Quantum computing

Quantum computing [20] is a process of manipulating quantum mechanical systems for information processing using superposition and entanglement. *Quantum computer* [5, 1] performs calculations quantum-mechanically using qubits, far more efficiently [9] than foreseeable classical computer. A qubit is a special system in quantum computing world with two degrees of freedom and state of a quantum system is represented using qubits with 0 and 1 in the superposition. The superposition is represented by state vector using $|\psi\rangle$ with $|0\rangle$ and $|1\rangle$ as basis states where $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$, where α and β are complex numbers. The basis states using which a quantum state is

represented are given by orthogonal vectors where, the inner product of two vectors $\langle 0|1\rangle = 0$. When a system is with multiple qubits, state of a composite quantum system is their tensor product $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$. Thus a quantum state is represented by basis states associated with some amplitude factor and these amplitudes are represented in terms of complex numbers (where the squares of the amplitudes are the probabilities of state being either 0 or 1 after measurement). Quantum computing can be considered as an inter disciplinary field where quantum mechanical concepts of physics applied in computer science for information processing. Using k-qubits, one can represent 2^k states. Fig. 1a is the representation of the quantum state of a qubit on a Bloch sphere [7].

A quantum computer operates on qubits using quantum gates [13], represented by a matrix that is applied to a state vector of the qubit. All the quantum operations are reversible and hence always given by unitary matrix $(U^T)^* = U^{-1}$. *Quantum circuit* is a sequence of operations in the form of gates on a quantum system as shown in Fig. 1b (b). The unitary operation transforms the qubit state to a different desired state. Any information out of a quantum system is extracted using physical measurement on the qubits. Thus, to look at any result in a quantum system, physical measurement brings it to the classical world.

3. Quantum computing approach for artificial neural networks (QC ANN)

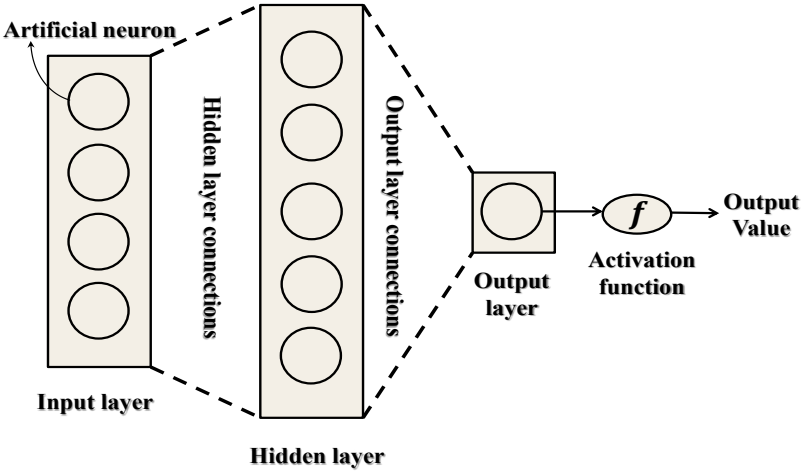


Fig. 2. Classical Artificial Neural Network (ANN) with artificial neurons as the nodes of the network

In this section, we first briefly describe the classical ANN before discussing QC ANN. In a classical ANN (Fig. 2), the *input layer* takes N-dimensional data onto the nodes called artificial neurons. These nodes are connected to the nodes in the *hidden layer* and the connections contain certain weights which are parameters that can be optimized. The *hidden layer* nodes are connected to the *output layer* again with weighted connections. Finally, an activation function is used to get an output value from the network. In the next step, the loss is calculated between actual output and the target output. The parameters are optimized through backpropagation until the loss minimizes between the actual output and the target output. The training process always depends on the dimension of the input and the size of the dataset. Training an ANN is a significant task where the parameters of the network are optimized by tuning the hyperparameters (parameters such as network depth, width) to avoid overfitting or underfitting of the model. The training task becomes complex with the increase in dimension and size of the data, as the size of parameters to be optimzed increases which require a lot of computation.

Quantum computing has the potential to handle such complex tasks that are intractable on a classical computer. The quantum computing approach for ANN (QC ANN) is given in Fig. 3. The artificial neurons in the *input layer* of classical ANN are replaced with qubits. The N-dimensional input data is encoded as a quantum state with the

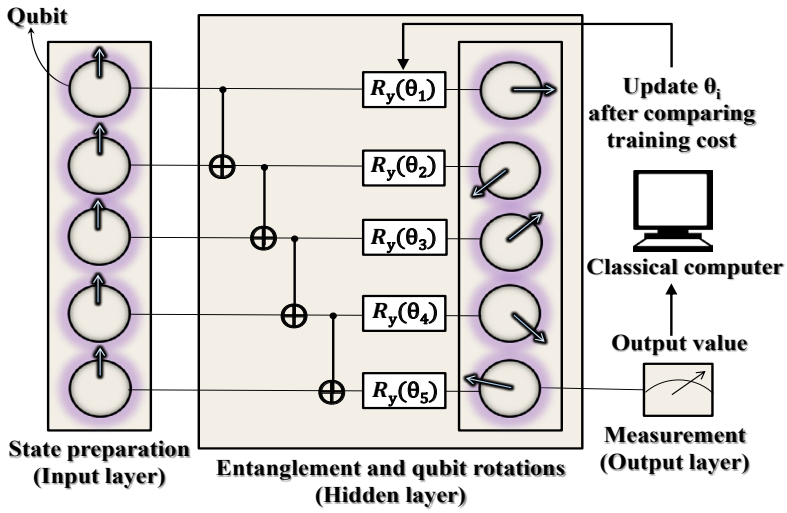


Fig. 3. QC ANN with qubits as nodes of the network using hybrid quantum-classical computation

superposition of 2^k states where k is the number of qubits. We use the amplitude embedding scheme [19] for the state preparation process. The *hidden layer* consists of connections between qubits for interaction as an *entanglement*. The states of the qubits are modified using gates with certain rotational parameters. Finally, we measure the state of a qubit for an output value. We compare the actual output value with the target value and optimize the rotational parameters of gates until we get the desired output after measurement. All the quantum operations in Fig. 3 can be executed on a near-term quantum computer and the optimization of rotational parameters of single-qubit gates can be done using a classical computer. This is a hybrid approach that uses both classical and quantum computers together for the binary classification task (details of the simulation are given in Section 4).

The learning processes of QC ANN is different from classical ANN. The QC ANN is implemented as a circuit on a quantum computer. The state preparation process allows encoding of the input data onto qubits as a quantum state. That is, the input data is embedded as amplitudes of different states in superposition. The qubits are now entangled in a way that, modification of state of one qubit influences the state of others. Then, the quantum state is modified with gates for single-qubit rotations through the hidden layers. In the *hidden layer* each qubit is rotated by a quantum gate and hence the state of the system changes. Finally, the state of a desired qubit is measured to obtain a value and compared with the target value to calculate the loss. The rotational parameters are modified until the quantum system reaches a state at which the desired qubit when measured, gives the target output value. The learning process here denotes that the quantum circuit learns the complex relationship between input and the output as rotational parameters of gates which transforms the state of the quantum system. Thus, the process of learning is entirely different from that in a classical ANN.

The advantages of QC ANN are:

- Using k qubits, 2^k attributes of the input data are encoded as a superposition of states onto a quantum computer. Consider a normalized input data instance of 32 values given as $x_1, x_2, x_3, ..., x_{32}$. Now the input data can be encoded as a quantum state (using 5 qubits) with superpositon of different states as $x_1 |00000\rangle + x_2 |00001\rangle, x_3 |00010\rangle +, ..., +x_{32} |11111\rangle$ using amplitude embedding.
- Using qubits as artificial neurons avoid the usage of nodes in the *hidden layers* as in classical ANN. Thus, the size of parameters to be optimized reduces, as there are no connections to be established between the layers.
- Quantum measurement itself acts as an activation function that saves the computational effort.

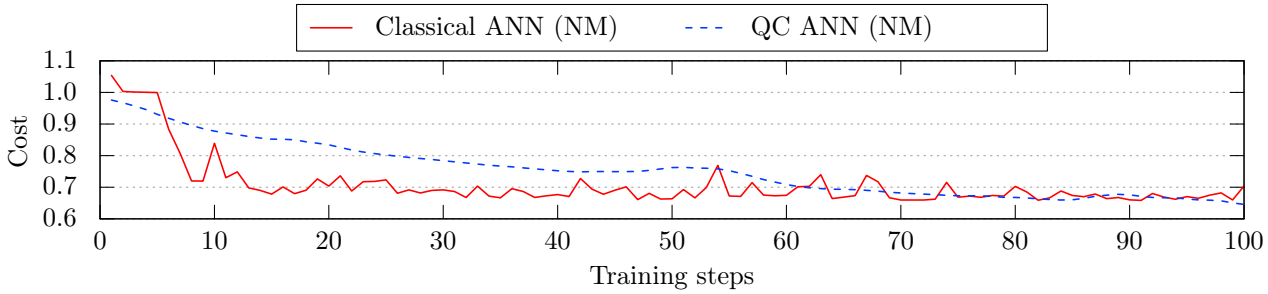


Fig. 4. Cost vs number of training steps of Classical ANN and QC ANN using NM

4. Experimental evaluation

In this section, we present the results of simulation performed to prove the efficiency of QC ANN. To implement classical ANN we used *keras* [4] library in *python*. QC ANN was experimented on a quantum simulator provided by *PennyLane* [2]. *PennyLane* also provides methods [17] for optimizing rotational parameters of quantum gates with hybrid quantum-classical computations. All the experiments are performed on *Wisconsin Breast Cancer Diagnosis* (WBCD) [23] dataset with two class labels.

Classical ANN consists of an *input layer* with 30 nodes, as the input dimension is 30 (it is not possible to reduce the number of nodes in the input layer) and two *hidden layers* with 9 and 3 nodes, respectively. The output layer is having 1 node as each data record is associated with a class label of binary value. Hence, the total number of parameters to be optimized are 300 $[(30 \times 9) + (9 \times 3) + (3 \times 1) = 300]$. We used *sigmoid* activation function for determining the class label.

Table 1. Details of hyperparameters used for training the networks in classical and quantum computing approach for ANN.

Type	Value
Cost function	Mean-squared error (MSE)
Optimization (SGD)	Nestrov momentum (NM) and Adaptive momentum estimation (Adam)
Learning rate	0.01
Momentum	0.9
Kernel initialization	Standard normal distribution

Table 2. Accuracy comparison of Classical ANN and QC ANN on WBCD dataset.

Type	Classical ANN	QC ANN
Training steps	100	100
Parameters trained	300	30
Training accuracy (426 records)	34.09% (NM) 98.12% (Adam)	83.33% (NM) 84.74% (Adam)
Validation accuracy (143 records)	39.86% (NM) 98.60% (Adam)	82.51% (NM) 83.21% (Adam)

The QC ANN is implemented using 5 qubits in the *input layer*. As there are 30 input values in the dataset, the values are embedded as amplitudes of quantum states. Therefore, it is required to use at least 5 qubits (as $2^5 = 32$). Amplitude embedding encoding scheme is used for the state preparation where the input data of 30 values are encoded as amplitudes of the quantum state with 32 superpositions. The other two values for the input are padded with constants to match the size of states in superposition. The amplitudes are represented by a state vector with a 32×1 dimension.

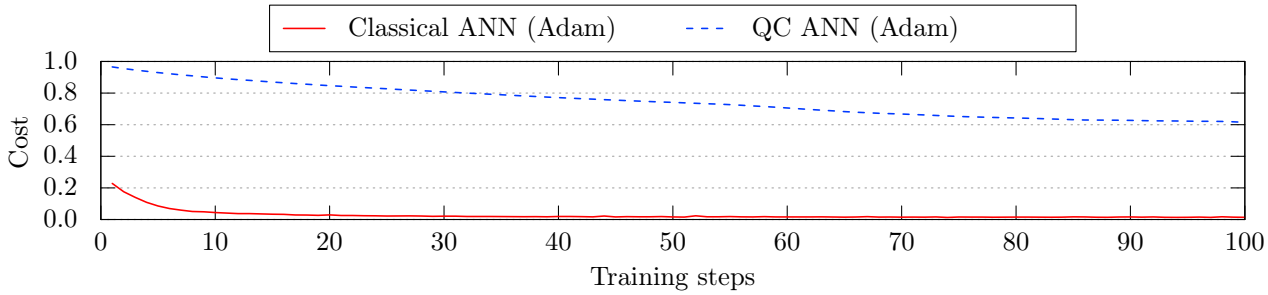


Fig. 5. Cost vs number of training steps of Classical ANN and QC ANN using Adam

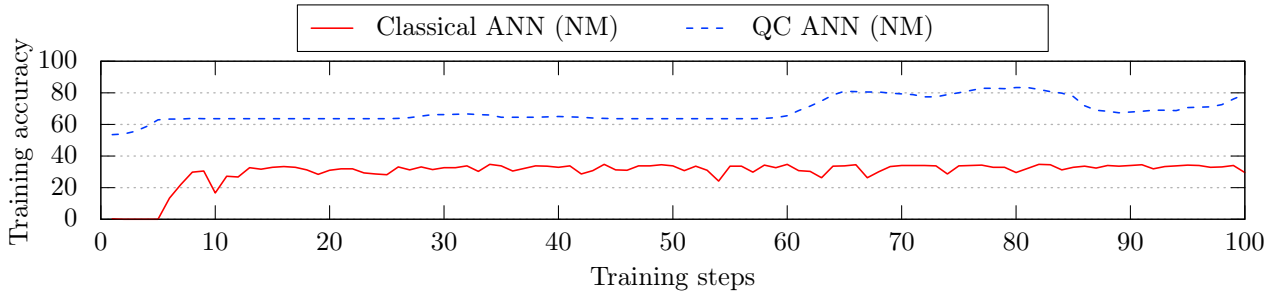


Fig. 6. Training accuracy comparison of Classical ANN and QC ANN using NM

The *hidden layer* consists of *Controlled NOT (CNOT)* gates for entanglement and $R_y(\theta)$ gates on each qubit. The $R_y(\theta)$ gate is a single-qubit rotation around $y-axis$ represented in the Bloch sphere (Fig. 1a) through an angle of θ (radians). We used six such hidden layers and the last qubit (Fig. 3) is measured using *Pauli-Z* (σ_z) measurement for an expectation value. A *Pauli-Z* (σ_z) measurement operator on basis states is represented as $\langle 0|\,\sigma_z\,|0\rangle$ and $\langle 1|\,\sigma_z\,|1\rangle$. As the operations $\langle 0|\,\sigma_z\,|0\rangle$ and $\langle 1|\,\sigma_z\,|1\rangle$ give binary values 1 and -1 respectively, this measurement is suitable for binary classification of the input data. The total number of parameters to be optimized in this approach is 30 as there are six hidden layers with five single-qubit rotations in each layer. The results show that QC ANN is outperforming the classical version of ANN for binary classification on *WBCD* dataset with a validation accuracy of 82.51% whereas the classical ANN is giving 39.86% validation accuracy using *Nesterov Momentum* (NM) [21]. However, Classical ANN is giving good accuracy over QC ANN using *Adaptive Momentum Estimation* (Adam) [11]. Both NM and Adam are Stochastic Gradient Descent (SGD) optimization techniques. Comparatively, QC ANN is able to perform well with very few parameters. Table 1 shows the details of hyperparameters used to train the models and comparison of accuracy is given in the Table 2.

4.1. Results and Discussion

In Fig. 4, comparison of the training cost for both classical ANN and QC ANN models using NM is provided. Mean-squared error is used as the cost function. The transition of the cost curve for the QC ANN is smooth, indicating that the model is able to learn the features from the data and can generalize the characteristics of the data efficiently in the model. However, the training cost for classical ANN is significantly less when compared to QC ANN using Adam indicating that tuning hyperparameters is an important task in training an artificial neural network.

In Figs. 6 and 7 the training accuracy of both the models is compared. Figs. 8 and 9 show the plots for validation accuracy. We can observe that with good training accuracy and validation accuracy, QC ANN is performing extremely well even when only fewer parameters are trained. Thus, using a quantum computer, we can train an ANN efficiently.

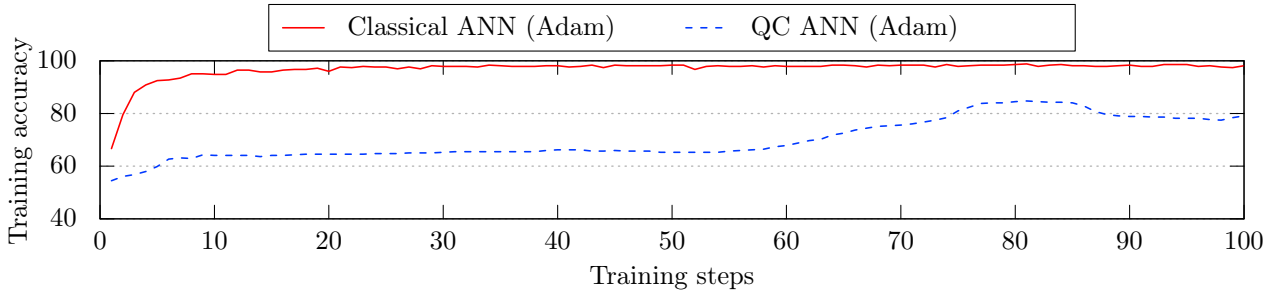


Fig. 7. Training accuracy comparison of Classical ANN and QC ANN using Adam

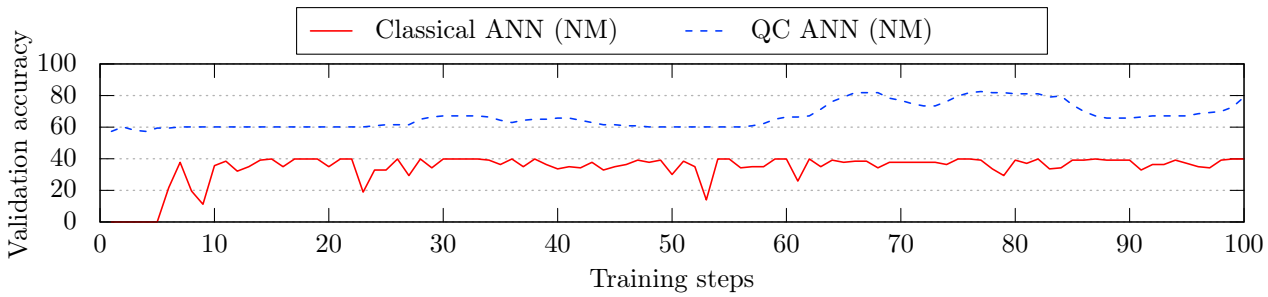


Fig. 8. Validation accuracy comparison of Classical ANN and QC ANN using NM

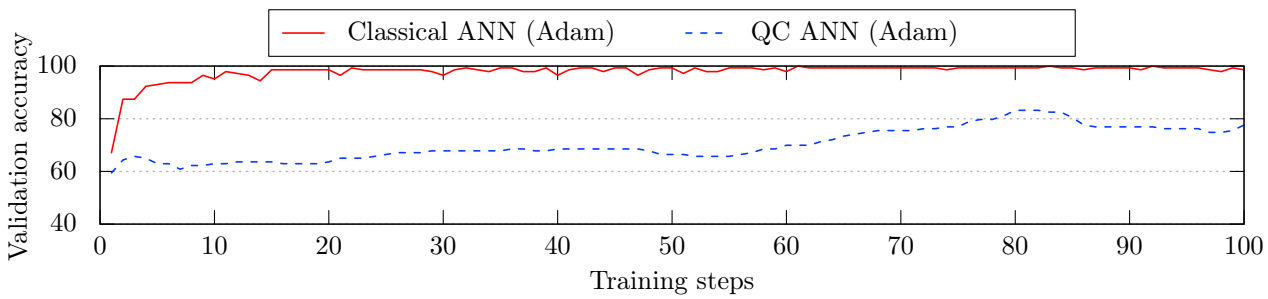


Fig. 9. Validation accuracy comparison of Classical ANN and QC ANN using Adam

5. Conclusion and Future Scope

Quantum computers have vast potential to solve many complex problems that are difficult for a classical computer to solve. In this work, we show that the quantum computing approach for training ANN on a quantum computer performs extremely well when compared with classical ANN. QC ANN used only a few parameters to learn the complex and non-linear patterns in data. Hence, it is computationally efficient. The training approach used in this work can also be extended to train deep neural network models for image classification. Also, the approach using a quantum computer in training a deep neural network is considered efficient in terms of computational space as we require only k qubits to represent 2^k input values of classical data. Thus, we can achieve good results using a few qubits and fewer parameters for optimization. The future scope of this work includes a study of the advantages of quantum computing on deep learning when working on datasets of large size and higher dimensions.

References

[1] Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G., Buell, D.A., et al., 2019. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510.

[2] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Killoran, N., 2018. Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968 .

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

[4] Chollet, F., et al., 2018. Keras: The python deep learning library. Astrophysics Source Code Library .

[5] Feynman, R.P., 1982. Simulating physics with computers. International journal of theoretical physics 21, 467–488.

[6] Fingerhuth, M., Babej, T., Wittek, P., 2018. Open source software in quantum computing. PloS one 13, e0208561.

[7] Gamel, O., 2016. Entangled bloch spheres: Bloch matrix and two-qubit state space. Physical Review A 93, 062320.

[8] Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., Green, A.G., Severini, S., 2018. Hierarchical quantum classifiers. npj Quantum Information 4, 65.

[9] Grover, L.K., 1996. A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA. pp. 212–219.

[10] Havlíček, V., Córcoles, A.D., Temme, K., Harrow, A.W., Kandala, A., Chow, J.M., Gambetta, J.M., 2019. Supervised learning with quantum-enhanced feature spaces. Nature 567, 209.

[11] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[12] LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444.

[13] Nielsen, M.A., Chuang, I.L., 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition. 10th ed., Cambridge University Press, New York, NY, USA.

[14] Preskill, J., 2018. Quantum computing in the nisq era and beyond. Quantum 2, 79.

[15] Reinsel, D., Gantz, J., Rydning, J., 2018. The digitization of the world: from edge to core. Framingham: International Data Corporation .

[16] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.

[17] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N., 2019. Evaluating analytic gradients on quantum hardware. Physical Review A 99, 032331.

[18] Schuld, M., Fingerhuth, M., Petruccione, F., 2017. Implementing a distance-based classifier with a quantum interference circuit. EPL (Europhysics Letters) 119, 60002.

[19] Schuld, M., Petruccione, F., 2018. Supervised Learning with Quantum Computers. volume 17. Springer International Publishing.

[20] Steane, A., 1998. Quantum computing. Reports on Progress in Physics 61, 117–173.

[21] Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of initialization and momentum in deep learning, in: International conference on machine learning, pp. 1139–1147.

[22] Tacchino, F., Macchiavello, C., Gerace, D., Bajoni, D., 2019. An artificial neuron implemented on an actual quantum processor. npj Quantum Information 5, 26.

[23] William H. Wolberg, W.N.S., Mangasarian, O.L., 1995. UCI machine learning repository.

[24] Zhou, L., Pan, S., Wang, J., Vasilakos, A.V., 2017. Machine learning on big data. Neurocomput. 237, 350–361.