Quantum Neural Network Parameter Estimation for Photovoltaic Fault Detection

Glen Uehara, Sunil Rao, Mathew Dobson, Cihan Tepedelenlioglu and Andreas Spanias
SenSIP Center, School of ECEE, Arizona State University

2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA) | 978-1-6654-0032-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/IISA52424.2021.9555558

***Abstract*— In this paper, we describe solar array monitoring using various machine learning methods including neural networks. We study fault detection using a quantum computer system and compare against results with a classical computer. We specifically propose a quantum circuit for a neural network implementation for Photovoltaic (PV) fault detection. The quantum circuit is designed for two qubits. Results and comparisons are presented for PV fault detection using a classical and quantum implementation of neural networks. In addition, simulations of a Quantum Neural Network are carried for a different number of qubits and results are presented for PV fault detection.**

I. Introduction

Wireless sensor networks and signal processing algorithms have been used for monitoring the voltage (V), current (I), irradiance and temperature of utility scale Photovoltaic (PV) arrays [1]. Originally, these studies explored the utility of standard statistical methods such as the Mahalanobis criterion [1] and later begun exploring the use of machine learning (ML) methods for PV fault detection. Fault detection using standard K-means methods has been reported in [2], [3], neural network methods (NN) in [4], [5], and positive unlabeled (PU) learning in [6]. Real-time study of faults on the DC side is enabled by wireless sensors and actuators. In particular, a smart monitoring device (SMD) has been developed in [7] and is shown in Figure 1. The SMD has sensors and actuators (relays) along with a microcontroller and an RF unit. It provides V-I, temperature, and irradiance data which can be shared on the Internet for analytics and control. It was shown that a solar array fitted with web-connected SMDs [3] forms an Internet of Things (IoT) network [8] where every solar panel can be accessed and provide data from remote locations. The SMD enables fault detection [9] and topology optimization [10]. With the use of measurements and relays, mal-functioning panels can be bypassed, and shaded panels can be reconnected from series to parallel to equalize radiance.

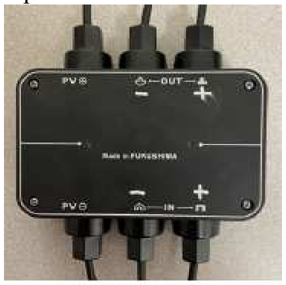

Figure 1 The smart monitoring device (SMD) for solar panel monitoring and control.

Topology reconfiguration using SMDs and machine learning has also been studied and results have been presented in [11]–[13].

In this paper, we explore machine learning algorithms for solar array monitoring and control. We study specifically neural network solutions for fault detection. We then implement a quantum computing [14] based neural network system. We present two implementations based on a design of a quantum circuit. The first hybrid quantum neural network (hybrid QNN1) is run with two qubit and four-qubit resolution. The second is an improved quantum circuit (hybrid QNN2) running with two-qubits. In our study , we show the accuracy, CPU run time, and the number of epochs used for QNN training.

The overall system used for solar array analytics, fault detection, and topology reconfiguration is shown in Figure 2. SMDs are installed one on each solar panel and provide data and the overall status on the PV array. The SMDs provide data via wireless connection and every panel has a MAC address and can be accessed after user authentication. These devices have relays and can form a switching matrix that can be used to form different panel connections.

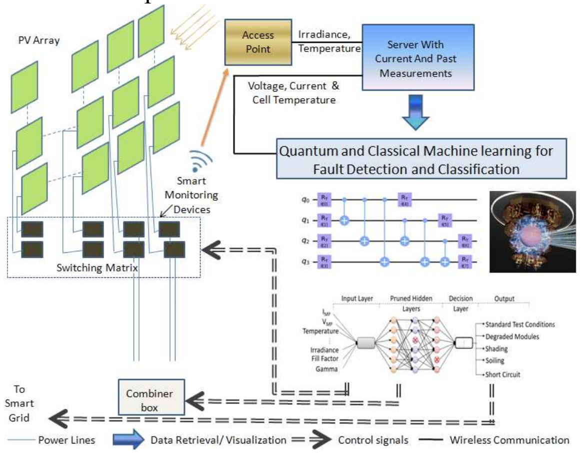

Figure 2 Smart solar array monitoring system integrated with quantum and classical machine learning fault diagnosis algorithms.

Fault Detection and Classification

The reliability of PV systems is essential in terms of maintaining the supply of power to the grid. Typically, solar panels are connected in series to form strings and then strings are connected in parallel. For example, in our solar array testbed facility at the ASU research park [3], we have 13 panels connected in series to form a string and then 8 such strings connected in parallel. PV faults studied include ground faults, series and parallel arc faults, soiling, and shading. Each of these faults has its own characteristics. Results and comparisons are presented for NN-based detection. The study on pruned NN is based Lottery Ticket Hypothesis optimization method [15].

The rest of the paper is organized as follows. Section II describes ML methods for fault detection. Section III presents work on the implementation of NN using quantum computing. Finally, section IV presents concluding remarks.

II. Machine Learning for PV Faults

Clustering PV faults using ML is useful for detecting and identifying the type of fault [16]. The problem has been handled before using statistical outlier detection techniques [2]. ML training requires sizeable data sets that cover array behavior for each fault condition. A comprehensive PV fault dataset does not currently exist and gathering real-time data for different fault conditions is difficult. For this reason, we used two data sets, namely the NREL data [17] and a synthetic dataset generated using the Sandia PV model [18]. The NREL PVWatts [18] includes data on 4 commonly occurring faults, as well as the standard test conditions of PV arrays. It covers the following categories: shaded, soiled, short-circuit, and degraded modules. The data was obtained for a period of one year and included irradiance, temperature, and maximum power measurements. In our classical computing fault detection simulations, we first examine neural networks and compare them against three supervised ML algorithms, namely, the Random Forest Classifier (RFC), the k-Nearest Neighbor (kNN), and the Support Vector Machine (SVM). We studied a set of 9-dimensional custom input features. The dataset contains a total of 22000 samples. We used a 22000 x 9 feature matrix to profile and compare the ML algorithms.

A. Neural Network Classifier for Fault detection

The NN architecture in Figure 3 uses the feature matrix as the input layer. It has five hidden layers consisting of six neurons each. Training is performed using standard backpropagation. The activation function in each neuron is a hyperbolic tangent, with a final SoftMax output activation function to estimate the fault type. Our simulations presented in the results section use 70% of the labeled data for training and the rest split evenly for validation and testing. Classification results on fault detection with comparisons to standard ML algorithms are presented in this section. Neural network simulations with quantum computing architectures are described in Section III.

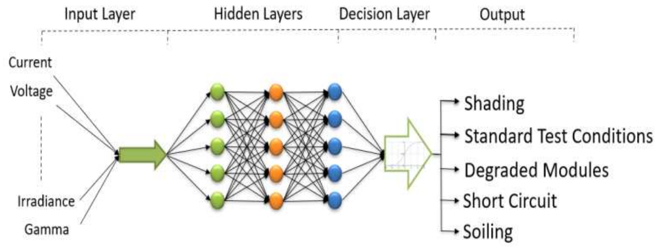

Figure 3 Neural Network for PV fault classification.

B. Random Forest Classifier

The Random Forest Classifier (RFC) is a classification algorithm based on an ensemble of decision trees. A decision tree is constructed by a set of input features from a randomly sampled batch of the dataset. RFC involves two hyper-parameters: the number of decision trees and the depth of the decision tree. RFCs are capable of modeling complex data sets and are robust to outliers [19].

B. The k-Nearest Neighbors

The k-nearest neighbor algorithm (kNN) [20] is a simple non-parametric classifier, where classification is based on local membership scores. In the training phase, a similarity measure for each data point with its closest *k* neighboring data points is stored. To classify a test sample, similarity scores between the test sample and all other data points are calculated, and the class label assigned is the label corresponding to the majority of *k* closest samples based on the similarity score.

C. The Kernel SVM

Kernel SVM is a soft margin classifier robust to outliers. Computing the soft margin classifier is equivalent to minimizing the loss function,

$$L_{svm} = \frac{1}{n}\left[\sum_{i=1}^{N} \max\left(0, 1 - y_i(w.\emptyset(x_i) - b)\right) + \lambda||w||^2\right], \quad (1)$$

where, λ is a hyper-parameter which regularizes the weights and $\emptyset(.)$ is the kernel function. The loss function in equation 1 can be reduced to a quadratic programming problem and solved by a convex solver. Common choices of kernel functions $\emptyset(.)$ are polynomial kernels, Gaussian radial basis kernels, and hyperbolic tangent kernels. The success of SVM depends on the appropriate choice of kernel.

D. Fault Detection Results and Comparisons

We compare the performance of the neural network fault detection against standard machine learning algorithms such as the RFC, SVM, and kNNs. These comparisons are developed on a classical computer. The RFC classifier was trained with 300 estimators with a depth of 50, the SVM was trained with radial basis kernel, and kNN with 30 nearest neighbors. We observe that techniques such as the RFC overfit the training data with a near-perfect training accuracy while the test accuracy is 86.32%. The SVM has a lower accuracy of 85.31% and a test accuracy of 83.29%. We also observe that the kNN gives a training accuracy of 87.15% and a test accuracy of 85.76%. Compared to these methods, the feed-forward NN provides the best results. In our studies, we obtained a training accuracy of 91.62% and a test accuracy of 89.34% using a feed-forward three-layer neural network. Furthermore fault detection research based on pruned and dropout neural networks [38] has also provided encouraging results. In addition to using classical computing methods for fault detection, Quantum computing (QC) implementations are also considered in this paper. Motivated by the promise of Quantum speeds, we attempted to design a quantum circuit for fault detection. The QC simulations are based on a two-qubit system and are described in the next section. Additional simulations with different numbers of qubits are also presented.

III. PV Fault Detection using a Quantum NN

In this section, we present preliminary work on Quantum NN (QNN) [21] parameter estimation that is based on state vector networks [22]. This system introduces a variational ansatz that can be used to represent the quantum ground states [23]. The implementation of a QNN system is currently constrained because of the limited availability and cost of a quantum computing system. A hybrid quantum-classical NN [24], [25] is proposed and evaluated for PV fault detection. The architecture for the hybrid quantum-classical NN is shown in Figure 4. This system uses the quantum circuit for the hidden layer of the neural network. For the PV Fault Detection, we run experiments only for fault/no-fault detection. This is because of constraints to work with manageable low-precision quantum circuit designs that have a limited number of qubits.

The hybrid QNN Design

The first step in building the hybrid QNN is to understand the detection problem. Quantum algorithms take advantage of the feature space by using the large dimensionality of quantum Hilbert space [26]. For this problem, we will operate the hybrid QNN by taking a quantum circuit with a feature map that would be analogous to the conventional SVMs.

The ML package is in the Qiskit SDK [27] and is called Qiskit Machine Learning [28]. Qiskit introduces a two-layer QNN that takes the parameterized operator to leverage a gradient framework to provide the backward pass. This two-layer hybrid QNN is designed and evaluated for PV fault/no-fault detection.

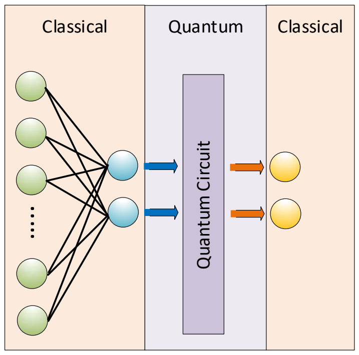

Figure 4 Hybrid quantum-classical System for QNN simulations.

The quantum expression of this quantum circuit is in the hidden layer. The circuit represents neurons that exploit quantum computing and encompass a nonlinear mapping capability. We developed preliminary hybrid QNN training simulation results, which we will validate later on an actual quantum computer with PV fault data.

The hybrid quantum-classical approach also allows for interfaces to existing ML toolkits. Included are a PyTorch [29] connection, a NN regressor, and a NN classifier. This quantum-enhanced design takes the current classical implementation and transforms different parts of the algorithm into the quantum platform.

A. The hybrid QNN for Fault Detection

In this paper, we present two hybrid QNN models for fault detection. The first, hybrid QNN1 is based on the combined circuits presented in Figure 7 and Figure 5. The second hybrid QNN2 is based on combined circuits in Figure 12 and Figure 13. The two hybrid QNN models are compared against the classical NN. The classical NN and hybrid QNN use the same Adam optimization [30] algorithm. However, the loss function differs between the two systems. The hybrid QNN uses the binary cross-entropy function [31], while the classical NN used a categorical cross-entropy function [32], [33]. To evaluate these systems we use the NREL dataset [17]. was chosen for both the classical NN described in Section II In our study, we used 80% and 20% of the data for training and testing, respectively.

We present results in terms of accuracy, CPU run time, training epoch count, and the number of qubits used.

B. The hybrid QNN1

The hybrid QNN1 for PV fault detection will use the two-layer QNN. To build this two-layer QNN, the feature map and ansatz quantum circuits are used. The quantum gates in the circuits will be used as the NN parameters [21].

We design the QNN using the *ZZFeaturemap* provided by the Qiskit Aqua [34] and the algorithms package in the Qiskit SDK. The feature map is used to select the number of qubits based on the input dimension of the data and its repetitions or the circuit depth. The initial design is based on a 2-qubit architecture in order to manage the computational complexity and time of the quantum simulations (Figure 7).

For the *ansatz* [35], [36], we choose the variational form as *RealAmplitudes*, from the Qiskit. An *ansatz* is the sequence of gates applied to specific wires of the system. For the hybrid QNN, we use these types of gates and the associated parameters for optimization and determination of the QNN weights. Figure 5 shows the quantum circuit for the two-qubit system. To complete the two-layer QNN, these two circuits are combined. As a note, to connect these two circuits, the same number of qubits between the feature map and *ansatz* circuit is needed.

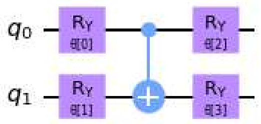

Figure 5 Quantum circuit of the hybrid QNN1 - RealAmplitudes [37].

To compare the hybrid QNN and the classical NN, we use the confusion matrix. The classical NN is evaluated as shown in Figure 6 and is found to have an accuracy of 95.29%.

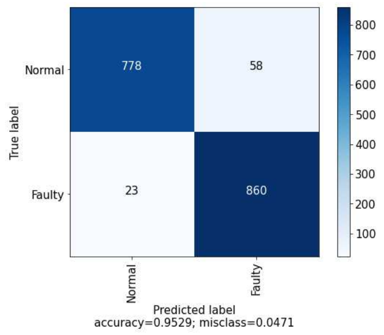

Figure 6 Confusion matrix of the classical NN.

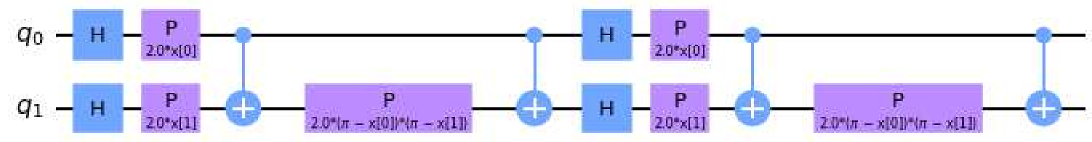

Figure 7 Quantum circuit of the feature map [34] used for the hybrid QNN1.

.

Our evaluation of the hybrid QNN1, starts by presenting one sample convergence curve with twenty-five epoch training as shown in Figure 8. The figure shows the binary cross-entropy learning curve for the model. As it is seen the algorithm converges.

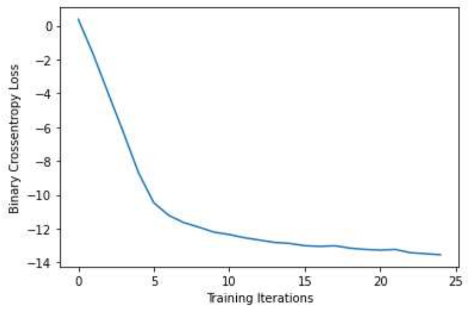

Figure 8 Binary Cross Entropy Learning Curve for 25 Epochs.

We present a similar confusion matrix for the fault detection on hybrid QNN1. This model has an accuracy of 87.8%, as seen in Figure 9.

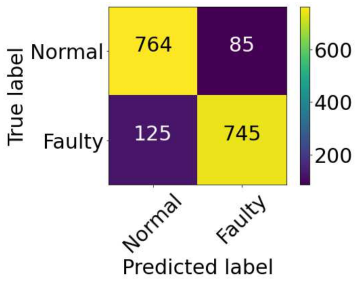

Figure 9 Confusion matrix of the hybrid QNN1 with 25 epoch training with 87.8% accuracy.

To explore further the quantum ML models, we attempt to improve the system by updating the quantum circuits as shown in the next subsection.

C. *Varying parameters in hybrid QNN1*

To determine if we can increase the accuracy, we began our experimentation with hybrid QNN1. For this, we explored changing the number of epochs (5, 10, 18, 25, 100) and varying the number of qubits (i.e., 2, 3, 4)

We experimented with different epoch counts. We present one of our findings where we increased our count to a hundred. We observe that the accuracy increases to 93.80%, as shown in Figure 10.

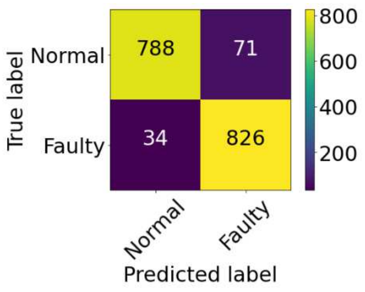

Figure 10 Confusion matrix of the hybrid QNN1 with 100 epoch training with 93.89 % accuracy.

Using the same feature map and *ansatz*, we updated the circuit to support additional qubits. Here, we note a smaller increase in accuracy as we increase the number of qubits. For example, for four qubits, our experiment resulted in an accuracy of 90.2%, as seen in Figure 11.

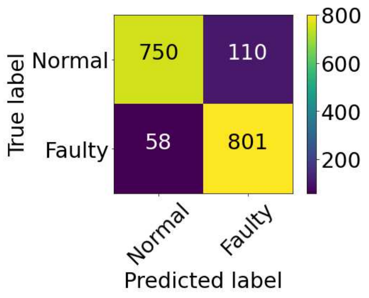

Figure 11 Confusion matrix of the hybrid QNN1 with 4 qubits with an accuracy of 90.2%

D. *The hybrid QNN2*

For hybrid QNN2, we examined ways to modify the feature map and *ansatz's* depth (or repetition) for this quantum circuit. Qiskit allows for both circuits to repeat. By repeating, additional gates are added, thereby increasing the depth of the circuit. These new quantum gates can be used for implementing NN parameters that are updated during the learning process. For example, in Figure 12, the *P*, *Ry*, and *Rz* gates represent the parameters that may be updated during the learning process.

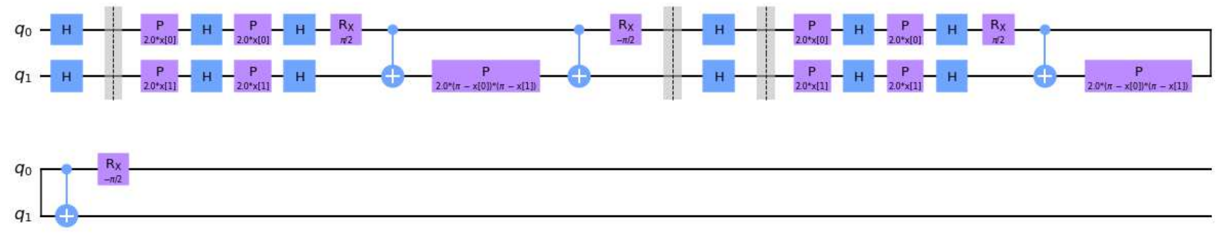

Figure 12 Updated feature map [34] (‘Z’, ‘X’, ‘ZY’) used hybrid QNN2.

We attempt to determine whether these additional parameters in the circuit have any impact on the learning process. We modified the feature map and the *ansatz* that was previously used for our hybrid QNN. For the feature map, a *PauliFeatureMap* was used (Figure 12). In this circuit, we created a gate combination of ‘*Z*’, ‘*X*’, and ‘*ZY’*. For the *ansatz* of the circuit in Figure 13, we updated it to use the built-in *EfficientSU2* as our function. As seen in Figure 13, we also repeated the base circuit to increase the number of parameters. This test is to determine whether these additional gates improve the learning rate. The goal was to increase the accuracy with fewer epochs. For this model validation, the epoch size was set to eighteen.

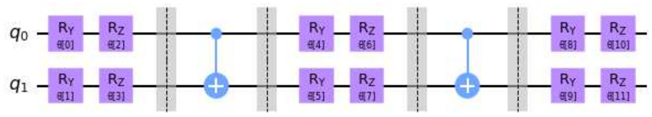

Figure 13 Updated *anastaz* used hybrid QNN2.

Running hybrid QNN2, we see in Figure 14 that our accuracy is now 90.5%.

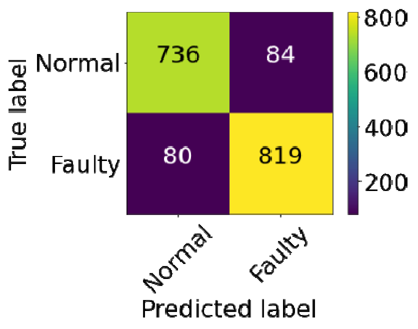

Figure 14 Confusion matrix of the hybrid QNN over 18 epochs.

E. The State Vector validation of hybrid QNN

As the last step in validating the hybrid QNN algorithm, we used state vector validation. This is a method to validate the algorithm without using a quantum simulator. In this simulation, we obtain a result of 93.8% accuracy.

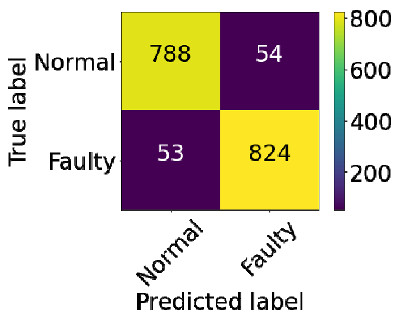

Figure 15 Confusion matrix of a hybrid QNN2 over 18 epochs.

F. Discussion of hybrid QNN

In our study, we are attempting to develop the best possible quantum model for fault detection given restrictions in the simulation execution and logistics. We examined the epoch length, increasing the number of quantum gates that can be used for representing parameters in a NN and the number of qubits. In our base circuit, we found that increase in epoch count helps improve accuracy. We also found that the increase in quantum gates helped reduce the epoch count. As we can see, epoch count is the major area to investigate given the difficult constraints of quantum circuit design and simulations. As we run with quantum simulators, the time for each epoch is an important factor in developing a circuit. The next step in our research are: a) investigate different QNN models to find an efficient balance of gates to be used as parameters for NN, b) adding GPU acceleration to the classical set, and c) possibly expand to add the complete fault classification tasks for the NREL dataset [17].

G. Fault Detection Results and Comparisons

The hybrid QNN models show similar detection test accuracy as the classical NN fault detection discussed in this research. The hybrid QNN designs presented in this paper are still not as computationally efficient as the classical NN in time or accuracy but show some promise. The following table presents initial results from our preliminary fault detection testing. As it can be seen, the accuracy of detection with the QNN is close to the classical NN detection presented in Section II.

In TABLE I. , we observe that increasing epochs has an impact on accuracy. However, the time to run this simulation is an important factor in building circuits to run on a quantum simulator. We next compare the increase in quantum gates versus the increase in qubits. Here, we find that the increase in quantum gates has a similar performance increase as adding additional qubits. Furthermore, in some cases we observed similar accuracy with fewer epochs.

TABLE I. FAULT DETECTION RESULTS AND COMPARISONS

Fault Detection Algorithm	Training Epoch	CPU Time	Detection Accuracy
Classical NN	300	20 sec	95.39%
Hybrid QNN1 (2 qubits)	25	10 hours	87.8%
Hybrid QNN1 (2 qubits),	100	~4 days	93.89%
Hybrid QNN1 (4 qubits),	25	~2 days	90.2%
Hybrid QNN2 (2 qubits)	18	8 hours	90.5%

These initial findings are important in determining the next steps in our research. The goal for fault detection is to build a quantum circuit with the appropriate number of quantum gates to increase accuracy yet keep the needed epoch count reasonable to run the quantum simulator.

IV. CONCLUSION

In this paper, we explored the use of NN for fault detection in an effort to improve performance and robustness in utility scale PV arrays. The methods presented are enabled by smart monitoring devices and neural networks that are applied for fault detection. The application of NN to fault detection provided higher accuracy in detecting and classifying faults relative to standard ML methods. In addition to fault detection using classical NN, we also presented the implementation of fault detection on a quantum system. Preliminary results with a two-qubit implementation provide modest fault detection relative to full precision classical computation. However, when qubits were added, we did not observe significant improvements in accuracy. We believe this is due to the quantum-related noise added when more qubits are used in the simulation. Furthermore, the QNN simulations required long computational times that prevented long training periods. Therefore, faster computing is needed in order to enable additional iterations (epochs) with the neural network. The intensive computing time required with the quantum simulator was in excess of four hours on an ordinary computer. Nevertheless, we anticipate elevating the accuracy significantly by gaining additional access to quantum simulators and designing higher resolution quantum circuits. To further improve accuracy and high qubit simulations of QNN, methods to reduce quantum measurement error have to be integrated.

ACKNOWLEDGEMENT

This research is supported in part by the NSF CPS Award number 1646542, the Quantum Computing NCSS SenSIP I/UCRC project, and the ASU SenSIP center.

REFERENCES

[1] H. Braun, S. T. Buddha, V. Krishnan, C. Tepedelenlioglu, A. Spanias, T. Takehara, T. Yeider, M. Banavar, and S. Takada, “Signal processing for solar array monitoring, fault detection, and optimization,” *Synth. Lect. Power Electron.*,Morgan and Claypool Publishers, vol. 4, pp. 1–95, 2012.

[2] R. Fazai, K. Abodayeh, M. Mansouri, M. Trabelsi, H. Nounou, M. Nounou, and G. E. Georghiou, “Machine learning-based statistical testing hypothesis for fault detection in photovoltaic systems,” *Sol. Energy*, vol. 190, pp. 405–413, 2019.

[3] S. Rao, D. Ramirez, H. Braun, J. Lee, C. Tepedelenlioglu, E. Kyriakides, D. Srinivasan, J. Frye, S. Koizumi, Y. Morimoto, and A. Spanias, “An 18 kW solar array research facility for fault detection experiments,” in *Proceedings of the 18th, MELECON 2016 IEEE, Limassol 2016.*

[4] S. Rao, A. Spanias, and C. Tepedelenlioglu, “Solar array fault detection using neural networks,” in *Proceedings - 2019 IEEE International Conference on Industrial Cyber Physical Systems, ICPS 2019*, 2019, pp. 196–200.

[5] E. Garoudja, A. Chouder, K. Kara, and S. Silvestre, “An enhanced machine learning based approach for failures detection and diagnosis of PV systems,” *Energy Convers. Manag.*, vol. 151, pp. 496–513, 2017.

[6] K. Jaskie, J. Martin, and A. Spanias, “Photovoltaic Fault Detection using Positive Unlabeled Learning,” Appl. Sci., no. Intelligent Fault Diagnosis of Power Systems, 2021.

[7] T. Takehara and S. Takada, “Photovoltaic panel monitoring apparatus.” *U.S. Patent No. 8,410,950.*, issued April 2, 2013.

[8] A. S. Spanias, “Solar energy management as an Internet of Things (IoT) application,” in *2017 8th International Conference on Information, Intelligence, Systems and Applications, IEEE IISA,* Larnaca, August, 2017.

[9] H. Braun, S. T. Buddha, V. Krishnan, A. Spanias, C. Tepedelenlioglu, T. Yeider, and T. Takehara, “Signal processing for fault detection in photovoltaic arrays,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2012, pp. 1681–1684.

[10] H. Braun, S. T. Buddha, V. Krishnan, C. Tepedelenlioglu, A. Spanias, M. Banavar, and D. Srinivasan, “Topology reconfiguration for optimization of photovoltaic array output,” Sustainable *Energy, Grids Networks (SEGAN)*, vol. 6, pp. 58–69, 2016.

[11] V. S. Narayanaswamy, R. Ayyanar, A. Spanias, C. Tepedelenlioglu, and D. Srinivasan, “Connection topology optimization in photovoltaic arrays using neural networks,” in *Proceedings - 2019 IEEE International Conference on Industrial Cyber Physical Systems, ICPS 2019*, May 2019, pp. 167–172.

[12] J. P. Storey, P. R. Wilson, and D. Bagnall, “Improved optimization strategy for irradiance equalization in dynamic photovoltaic arrays,” *IEEE Trans. Power Electron.*, vol. 28, no. 6, pp. 2946–2956, 2013.

[13] M. Jazayeri, K. Jazayeri, and S. Uysal, “Adaptive photovoltaic array reconfiguration based on real cloud patterns to mitigate effects of non-uniform spatial irradiance profiles,” *Sol. Energy*, vol. 155, pp. 506–516, 2017.

[14] G. Uehara, A. Spanias, and W. Clark, “Quantum Computing Algorithms for Machine Learning and Signal Processing.” *IEEE IISA 2021*, Crete, July 2021.

[15] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *7th Int. Conf. Learn. Represent. ICLR 2019*, 2019.

[16] S. Rao, S. Katoch, V. Narayanaswamy, G. Muniraju, C. Tepedelenlioglu, A. Spanias, P. Turaga, R. Ayyanar, and D. Srinivasan, “Machine Learning for Solar Array Monitoring, Optimization, and Control,” *Synth. Lect. Power Electron.*, Morgan and Claypool Publishers, vol. 7, no. 1, pp. 1–91, 2020.

[17] A. Dobos, “PVWatts version 1 technical reference,” *Nrel/Tp*, no. October, p. 1 online resource (8 pages), 2013, [Online]. Available: http://purl.fdlp.gov/GPO/gpo51913.

[18] J. Peng, L. Lu, H. Yang, and T. Ma, “Validation of the Sandia model with indoor and outdoor measurements for semi-transparent amorphous silicon PV modules,” *Renew. Energy*, vol. 80, pp. 316–323, 2015.

[19] T. K. Ho, “Random decision forests,” in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1995, vol. 1, pp. 278–282.

[20] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992.

[21] Z. Jia, B. Yi, R. Zhai, Y. Wu, G. Guo, and G. Guo, “Quantum Neural Network States: A Brief Review of Methods and Applications,” *Adv. Quantum Technol.*, vol. 2, no. 7–8, p. 1800077, 2019.

[22] C. Cortes and V. Vapnik, “Support-vector networks,” in *Machine learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[23] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, and others, “Qiskit: An open-source framework for quantum computing,” *Accessed on: Mar*, vol. 16. 2019. [Online].

[24] X. Hong and C. Maojun, “Hybrid quantum neural networks model algorithm and simulation,” in *5th International Conference on Natural Computation, ICNC 2009*, 2009, vol. 1, no. 1, pp. 164–168.

[25] M. A. Metawei, H. Said, M. Taher, H. Eldeib, and S. M. Nassar, “Survey on Hybrid Classical-Quantum Machine Learning Models,” in *Proceedings of the 2020 IEEE International Conference on Communications, Computing, Cybersecurity, and Informatics, CCCI 2020*, 2020, pp. 1–6.

[26] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019.

[27] *Open-source quantum development*. Qiskit. (n.d.). https://qiskit.org/.

[28] Qiskit, “Introducing Qiskit Machine Learning,” *Medium*, 2021. https://medium.com/qiskit/introducing-qiskit-machine-learning-5f06b6597526.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, *et al.*, “PyTorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32. 2019.

[30] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” Dec. 2015. Accessed: Jun. 19, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9.

[31] A. Creswell, K. Arulkumaran, and A. A. Bharath, “On denoising autoencoders trained to minimise binary cross-entropy,” Aug. 2017, Accessed: Jun. 07, 2021. [Online]. Available: http://arxiv.org/abs/1708.08487

[32] R. Atienza, *Advanced Deep Learning with Keras*. 2018. Accessed: Jun. 21, 2021. [Online]. Available: https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras.

[33] A. Kumar, “Keras - Categorical Cross Entropy Loss Function - Data Analytics,” *Data Analytics*, 2020. https://vitalflux.com/keras-categorical-cross-entropy-loss-function/ (accessed Jun. 21, 2021).

[34] “Feature Maps (qiskit.aqua.components.feature_maps) - Qiskit 0.26.2 documentation.” https://qiskit.org/documentation/apidoc/qiskit.aqua.components.feature_maps.html.

[35] J. Hermann, Z. Schätzle, and F. Noé, “Deep-neural-network solution of the electronic Schrödinger equation,” *Nat. Chem.*, vol. 12, no. 10, pp. 891–897, 2020.

[36] J. Bausch and F. Leditzky, “Quantum codes from neural networks,” *New J. Phys.*, vol. 22, no. 2, p. 23005, 2020d.

[37] Qiskit, “Basic Qiskit Syntax.” https://qiskit.org/textbook/ch-appendix/qiskit.html.

[38] S. Rao, G. Muniraju, C. Tepedelenlioglu, D. Srinivasan, G. Tamizhmani and A. Spanias, "Dropout and Pruned Neural Networks for Fault Classification in Photovoltaic Arrays," *IEEE Access*, 2021.