Solar Array Fault Detection Using Classical and Quantum Support Vector Classifiers

Niki Kyriacou
Department of Physics
Arizona State University
Tempe, USA
nkyriaco@asu.edu

Glen Uehara
Department of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, USA
guehara@asu.edu

Sameeksha Katoch
Department of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, USA
skatoch1@asu.edu

Andreas Spanias
Department of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, USA
spanias@asu.edu

Lenos Hadjidemetriou
KIOS Center
University of Cyprus
Nicosia, Cyprus
hadjidemetriou.lenos@ucy.ac.cy

Maria Michael
KIOS Center
University of Cyprus
Nicosia, Cyprus
mmichael@ucy.ac.cy

Abstract— **Photovoltaic arrays require real-time monitoring and maintenance for optimal performance. Detection and identification of faults are critical to this maintenance. The four most commonly occurring faults are shading, degraded modules, soiling, and short circuits. This paper compares the effectiveness of classical and quantum support vector classifiers in identifying these faults.**

***Keywords*— photovoltaic systems, fault detection, quantum machine learning, support vector classifier**

I. Introduction

As the renewable energy industry expands, the presence of photovoltaic systems is becoming more common. To ensure that investments in PV systems are protected, it is necessary to have systems in place to monitor and maintain their performance. One major part of this maintenance is fault detection. The most common faults in PV systems are shading, degraded modules, soiling, and short circuits [1].

SenSIP Lab has done several studies demonstrating the effective use of advanced neural networks to accurately detect and identify these commonly occurring faults to a high degree of accuracy [2-7]. The primary indicators used in this fault detection were voltage, current, temperature, and irradiance [2].

Figure 1: Photos of PV arrays containing the four most common faults [1].

More recently, SenSIP Lab has published a study comparing the effectiveness of PV fault detection using classical neural networks and simulated quantum neural networks [6]. Although the study did not demonstrate significant improvement in accuracy using quantum neural networks, it was posited that developing higher resolution quantum circuits could potentially provide an advantage over classical neural networks [8].

Additional research into quantum machine learning has demonstrated that QML algorithms are well suited to solving decision making problems and identifying patterns [9-11]. Furthermore, it has long been established that a support vector machine (SVM) could be used for classification on a quantum computer [12]. More recent work has even demonstrated that quantum SVMs have outperformed classical ones [13].

This research investigates the training of simulated quantum support vector classifiers using the Qiskit platform and comparing the results to those of classical support vector classifiers.

II. Data

The data used in this research was obtained from the previous study done by SenSIP Lab published in *Machine learning for solar array monitoring, optimization, and control* [1]. This is an evenly distributed classification dataset containing five panel classes associated with different data features. The five classes are soiled, shaded, degraded, short circuit (SC), and standard test conditions (STC).

The specified data includes ten major features which were used to for fault detection. These features include power output, temperature, irradiance, fill factor, open circuit voltage (V_{OC}), short-circuit current (I_{SC}), maximum voltage (V_{MP}), maximum power (P_{MP}), maximum current (I_{MP}), and gamma (γ) which is the ratio of power over irradiance.

III. Classical Algorithms

The first part of this project trained SKLearn's Support Vector Classifier (SVC) to categorize the data into one of the five classes [14]. Before using the data to train the model, the data underwent pre-processing. SKLearn's StandardScaler was used to standardize the dataset and SKLearn's univariate feature selection was used to select the four most significant features [15]. Finally, the data was divided into an 80/20 train/test split.

A SVC creates a hyperplane to separate data belonging to different classes. SVCs utilize kernel functions to assist with classification. SKLearn has four main kernel functions that can be selected: linear, polynomial, radial basis function (RBF), and sigmoid.

Simulations were run with each of the four kernels, and the kernels were evaluated on the basis of precision, recall, and F-score. The RBF kernel performed the best of the four.

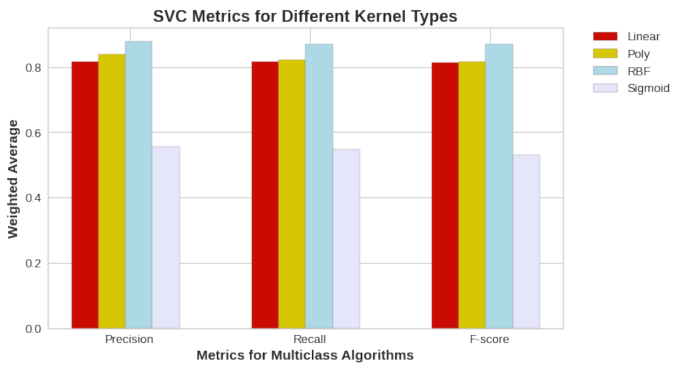
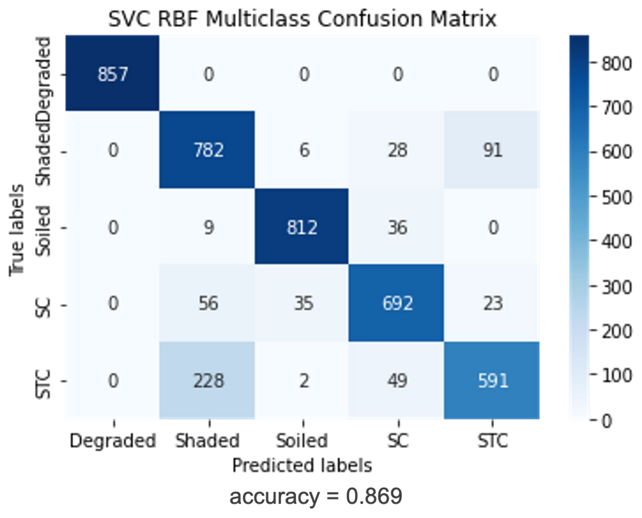


Figure 2: Graph depicting precision, recall, and F-score values for four different kernel types for a multi-class SVC.

The SVC was then trained using the RBF kernel to sort the data into one of the five classes.



True \ Predicted	Degraded	Shaded	Soiled	SC	STC
Degraded	857	0	0	0	0
Shaded	0	782	6	28	91
Soiled	0	9	812	36	0
SC	0	56	35	692	23
STC	0	228	2	49	591

Figure 3: Confusion matrix depicting the results of the multi-class SVC training.

After training the SVC, it was able to predict the class of that data with almost 87% accuracy. This score would serve as the metric for the quantum machine learning algorithm to either meet or exceed.

However, due to limited availability of processing power, the quantum machine learning algorithm would only be able to perform binary classification. Therefore, to establish another basis of comparison, another SVC was trained to separate the data into two classes: faulty and not faulty.

The same process was followed as the multi-class SVC, and simulations were run with all four kernels. The RBF kernel performed the best of the four again, although by a much smaller margin this time.

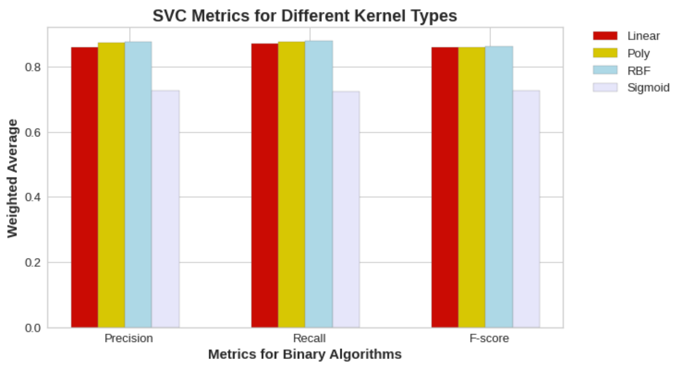
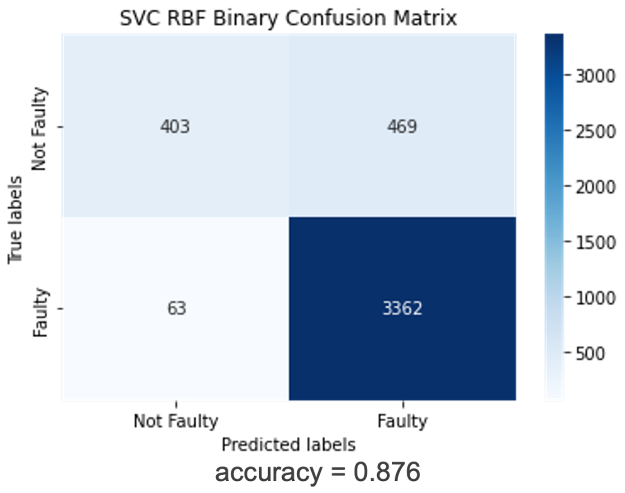


Figure 4: Graph depicting precision, recall, and F-score values for four different kernel types for a binary SVC.

After training the SVC, it was able to classify the data as faulty or not faulty with an accuracy of almost 88%.



True \ Predicted	Not Faulty	Faulty
Not Faulty	403	469
Faulty	63	3362

Figure 5: Confusion matrix depicting the results of the binary SVC training.

Consequently, the aim for the quantum machine learning algorithm was to meet or exceed a classification accuracy of 87-88%.

IV. Quantum Algorithms

The second part of this project used Qiskit software to simulate quantum machine learning algorithms. The intention was to evaluate whether there could be an empirical quantum advantage.

A. Quantum Support Vector Classifier

Before the data could be used to train the model, it underwent pre-processing. SKLearn's MinMaxScaler was used to standardize the dataset, and SKLearn's univariate feature selection was used to select the two most significant features. Due to limited processing power, a function was included to randomly select half of the dataset to be used in the algorithm. Finally, the data was divided into an 80/20 train/test split.

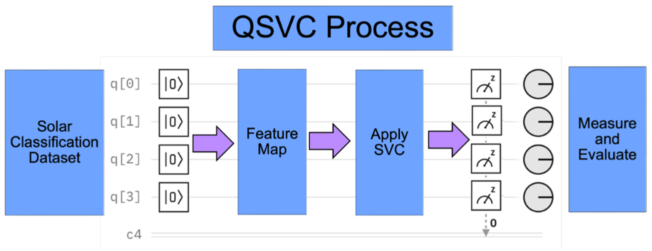


Figure 6: Depiction of the process to use the quantum support vector classifier.

To use the dataset in a quantum machine learning algorithm, a feature map was used to map the data to the quantum Hilbert space [16].

This algorithm used Qiskit's ZZFeatureMap with two reps and linear entanglement [17]. Two qubits were used for this algorithm due to limited computer processing power.

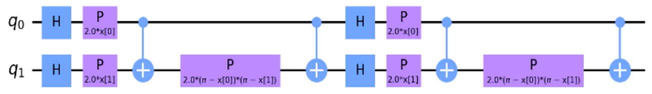


Figure 7: Example of a two-qubit feature map used in Qiskit.

Once the feature map was applied, the quantum support vector classifier (QSVC) was trained and then the results were measured and evaluated. Qiskit's QSVC algorithm is an extension of SKLearn's classical SVC and therefore also works with its methods and metrics for evaluation [18].

The QSVC was first trained to perform binary classification of faulty or not faulty. Qiskit's Statevector simulator was used because it had the fastest processing speed, and the algorithm was run with 1024 shots.

After training, the QSVC was able to classify the data as faulty or not faulty with an accuracy of approximately 90%.

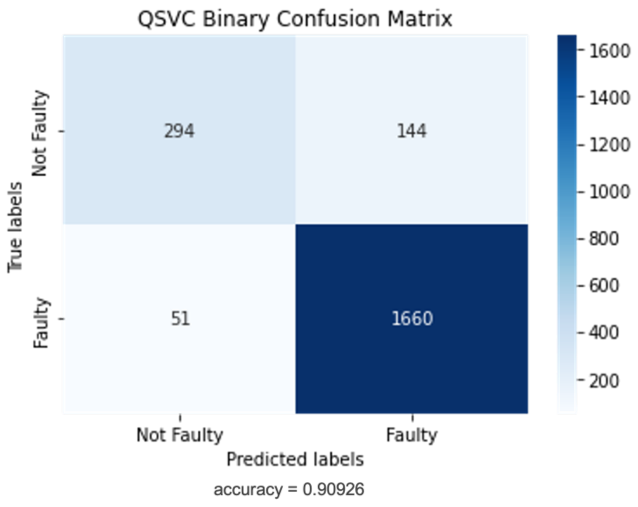


Figure 8: Confusion matrix depicting the results of the binary QSVC training.

Since the QSVC was only able to perform binary classification, the same parameters were used for each of the five original classes of shaded, degraded, soiled, SC, or STC.

The QSVC was able to classify the degraded panels with approximately 99% accuracy, the shaded panels with approximately 91% accuracy, the soiled panels with approximately 97% accuracy, the SC panels with approximately 95% accuracy, and the STC panels with approximately 91% accuracy.

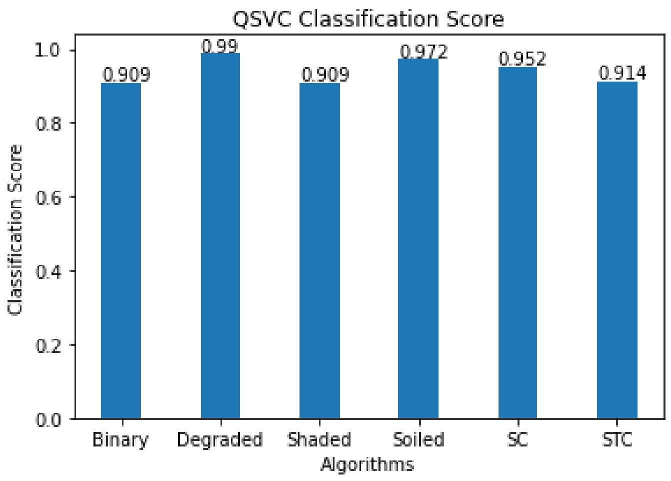


Figure 9: Graph depicting classification score for the six different binary QSVC results.

Qiskit's QSVC algorithm provided encouraging results. However, the long processing times and the inability to use the entire dataset led to the exploration of another Qiskit algorithm.

B. *Quantum Support Vector Classifier Pegasos*

The next algorithm this project explored was Qiskit's QSVC Pegasos, which was expected to train faster than the QSVC algorithm [19]. The algorithm was based on the paper by Shalev-Shwartz et al. titled *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM* [20].

The data pre-processing for this algorithm included the use of SKLearn's MinMaxScaler to standardize the dataset, and the use of SKLearn's univariate feature selection to select the four most significant features. It was again necessary to include a function was to randomly select half of the dataset to be used in the algorithm. Finally, the data was divided into an 80/20 train/test split.

This algorithm used Qiskit's ZFeatureMap with two qubits and two reps [21]. QSVC Pegasos was also run on the Statevector simulator. The standard value of 100 was used for the number of steps (τ) and a value of 1000 was used for the regularization hyperparameter (C) [19].

Just as with the QSVC algorithm, the QSVC Pegasos algorithm was first trained for the binary classification of fault or not faulty and then trained to perform binary classification for each of the original five classes.

After training, the algorithm was able to classify the data as faulty or not faulty with an accuracy of approximately 83%.

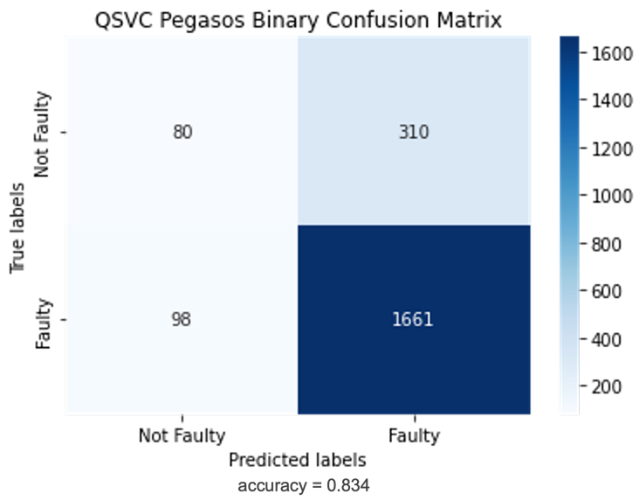


Figure 10: Confusion matrix depicting the results of the binary QSVC Pegasos training.

Additionally, the QSVC Pegasos algorithm was able to classify the degraded panels with approximately 82% accuracy, the shaded panels with approximately 85% accuracy, the soiled panels with approximately 69% accuracy, the SC panels with approximately 53% accuracy, and the STC panels with approximately 81% accuracy.

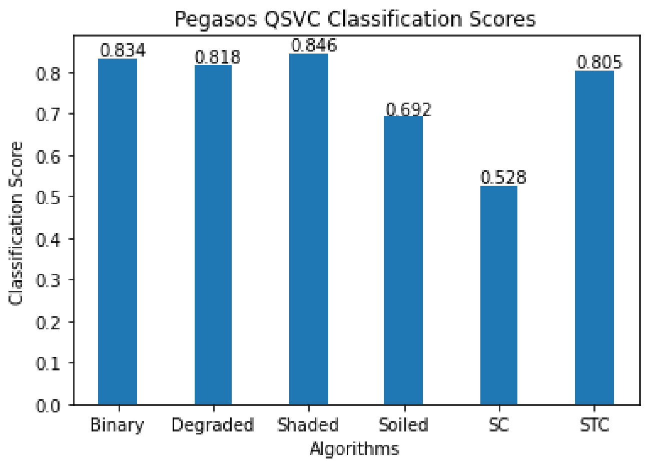


Figure 10: Graph depicting classification score for the six different binary QSVC Pegasos results.

V. Conclusion and Future Work

Overall, the results were mixed. The performance of the algorithms from best to least accuracy was the QSVC algorithm, the classical SVC algorithm, and finally the QSVC Pegasos algorithm.

It is important to note that both quantum machine learning algorithms were only run with random selections of half of the original dataset due to limitations in computer processing power. Furthermore, the quantum algorithms were limited to binary classification while the classical SVC allowed the use of multiple classes.

For future work, it would be beneficial to run the algorithms on more powerful computers to evaluate the change in performance when using the entire dataset. Additionally, since Qiskit only simulates the use of a quantum computer, it would be useful to compare the results using an actual quantum computer. Finally, the dataset used was simulated, evenly distributed, and well-behaved. It would be advantageous to evaluate these algorithms with real world data to better understand their performance.

Acknowledgment

This project was sponsored by National Science Foundation Award 1854273.

References

[1] 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,” *Synthesis Lectures on Power Electronics*, vol. 7, no. 1, pp. 1–91, 2020.

[2] S. Rao, A. Spanias, C. Tepedelenliglu, "Solar Array Fault Detection using Neural Networks", IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Taipei, May 2019.

[3] 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.

[4] V. Narayanaswamy, R. Ayyanar, A. Spanias, C. Tepedelenlioglu, "Connection Topology Optimization in PV Arrays using Neural Networks'," IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Taipei, May 2019.

[5] K. Jaskie, J. Martin, and A. Spanias, “PV Fault Detection using Positive Unlabeled Learning,” Applied Sciences, vol. 11, Jun. 2021.

[6] H. Braun, S. Buddha, V. Krishnan, C. Tepedelenlioglu, A. Spanias, S.i Takada, T. Takehara, M. Banavar, and T. Yeider., Signal Processing for Solar Array Monitoring, Fault Detection, and Optimization, Synthesis Lectures on Power Electronics, Morgan & Claypool, Book, 1-111 pages, ISBN 978-1608459483, Sep. 2012.

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

[8] G. Uehara, S. Rao, M. Dobson, C. Tepedelenlioglu and Andreas Spanias, "Quantum Neural Network Parameter Estimation for Photovoltaic Fault,” *Proc. IEEE IISA 2021*, July 2021.

[9] R. Divya and J. Dinesh Peter, "Quantum Machine Learning: A comprehensive review on optimization of machine learning algorithms," *2021 Fourth International Conference on Microelectronics, Signals & Systems (ICMSS)*, 2021, pp. 1-6, doi: 10.1109/ICMSS53060.2021.9673630.

[10] H. Yano, Y. Suzuki, K. M. Itoh, R. Raymond and N. Yamamoto, "Efficient Discrete Feature Encoding for Variational Quantum Classifier," in *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1-14, 2021, Art no. 3103214, doi: 10.1109/TQE.2021.3103050.

[11] Hong, Y.Y. and Pula, R.A., 2022. Methods of photovoltaic fault detection and classification: A review. *Energy Reports*, *8*, pp.5898-5929.

[12] P. Rebentrost, M. Mohseni and S. Lloyd, "Quantum Support Vector Machine for Big Data Classification", 2013. [Online]. Available: https://doi.org/10.1103/PhysRevLett.113.130503.

[13] S. Kavitha and N. Kaulgud, "Quantum machine learning for support vector machine classification", *Evolutionary Intelligence*, 2022. Available: 10.1007/s12065-022-00756-5.

[14] "sklearn.svm.SVC", *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[15] "sklearn.feature_selection.SelectKBest", *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html.

[16] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *arXiv.org*, 19-Mar-2018. [Online]. Available: https://arxiv.org/abs/1803.07128.

[17] "ZZFeatureMap — Qiskit 0.37.1 documentation", *Qiskit.org*. [Online]. Available: https://qiskit.org/documentation/stubs/qiskit.circuit.library.ZZFeatureMap.html.

[18] "QSVC — Qiskit Machine Learning 0.4.0 documentation", *Qiskit.org*. [Online]. Available: https://qiskit.org/documentation/machine-learning/stubs/qiskit_machine_learning.algorithms.QSVC.html.

[19] "Pegasos Quantum Support Vector Classifier — Qiskit Machine Learning 0.4.0 documentation", *Qiskit.org*. [Online]. Available: https://qiskit.org/documentation/machine-learning/tutorials/07_pegasos_qsvc.html.

[20] S. Shalev-Shwartz, Y. Singer, N. Srebro and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM", *Mathematical Programming*, vol. 127, no. 1, pp. 3-30, 2010. Available: 10.1007/s10107-010-0420-4.

[21] "ZFeatureMap — Qiskit 0.37.1 documentation", *Qiskit.org*, 2022. [Online]. Available: https://qiskit.org/documentation/stubs/qiskit.circuit.library.ZFeatureMap.html.