Quantum Convolutional Neural Network-based Online Malware File Detection for Smart Grid Devices

Alve Rahman Akash, BoHyun Ahn, Alycia Jenkins, Ameya Khot, Lauren Silva, Hugo Tavares-Vengas, and Taesic Kim*
[2]Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX, 78363 USA
{alve.akash; bohyun.ahn; alycia.jenkins; ameya.khot; lauren.silva; hugo.tavares-venegas}@students.tamuk.edu, taesic.kim@tamuk.edu

***Abstract*—Cybersecurity concerns have arisen due to extensive information exchange among networked smart grid devices which also employ seamless firmware update. An outstanding issue is the presence of malware-injected malicious devices at the grid edge which can cause severe disturbances to grid operations and propagate malware on the power grid. This paper proposes a cloud-based, device-specific malware file detection system for smart grid devices. In the proposed system, a quantum-convolutional neural network (QCNN) with a deep transfer learning (DTL) is designed and implemented in a cloud platform to detect malware files targeting various smart grid devices. The proposed QCNN algorithm incorporates quantum circuits to extract more features from the malware image files than the filter in conventional CNNs and the DTL method to improve detection accuracy for different types of devices (e.g., processor architecture and operating systems). The proposed algorithm is implemented in the IBM Watson Studio cloud platform that utilizes IBM Quantum processor. The experimental results validate that the proposed malware file detection method significantly improves the malware file detection rates compared to the conventional CNN-based method.**

Keywords—cybersecurity, deep transfer learning, cybersecurity, malware detection, quantum convolutional neural network, smart grid devices

I. INTRODUCTION

The electric power grid has been transitioning into a smart grid with advanced networks and computational systems to provide enhancements for core grid needs such as situational awareness, optimal and resilient operations, and maintenance. Furthermore, newly developed smart grid devices (e.g., smart meters [1] and inverters [2]), software-defined networks [3], and microgrids [4] that update software to adapt to new workloads and demands enable solving grid problems at grid edge (i.e., software-defined smart grid [5]). However, cybersecurity concerns arise due to extensive information exchange among the networked devices, which may also employ seamless firmware update functions. Therefore, the attack surface of the smart power grid has been significantly expanded [6], [7].

An outstanding threat is sophisticated attackers who keep trying to attack the main control center using malware (e.g., Stuxnet [8], Ukraine's power grid attacks [9], SolarWinds attack [10], and ransomware attack on Colonial Pipeline [11]). Recently, adversaries to target networked embedded systems (e.g., Conti ransomware attacked wind turbines [12]). It is anticipated that malware-injected malicious devices at the grid edge can cause severe disturbances to grid operations and propagate malware on the power grid [13], which has been less studied.

Recently, researchers have begun to explore the malware security of smart grid devices using machine learning (ML). Malicious smart inverter controller firmware modifications could be detected by machine learning (ML) algorithms that are trained and validated by data acquired from custom-built hardware performance counters (HPCs) using assembly firmware files [14]. However, a device-specific disassembler is necessary. The authors proposed a convolutional neural network (CNN)-based ransomware detection using 2-D grayscale image files converted from binary files without using a disassembler [15]. To detect smart inverter-specified malware types, the authors proposed a deep transfer learning (DTL)-based CNN method as a host-based malware file detection solution [16]. To find more malware detection methods, interest readers may read a survey paper [17].

Meanwhile, the advent of quantum computers has led to develop quantum machine learnings (QMLs) which integrates quantum algorithms into classical ML models to improve the performance of ML models or address computing power issues, especially when training the convention deep learning models [18]. In [19], a quantum CNN (QCNN) was developed to classify image files. A series of quanvolutional layers consisting of quantum circuits in a gate model-based quantum computer was used to extract hidden features from the images.

This paper is an extended work of [16] to develop a cloud-based, device-specific malware file detection system utilizing QCNN and DTL techniques for smart grid devices. The proposed Q-CNN algorithm utilizes quantum circuits to extract more features from the malware image files than the filter used in the CNN [16]. The proposed algorithm is validated in the IBM Watson Studio cloud platform that utilizes an IBM-Q quantum processor. The experimental results show that the proposed method significantly improves the detection accuracy compared to the conventional CNN model.

II. RELATED WORK: SMART GRID DEVICE MALWARE THREAT

As a case study on malware file detection for smart grid devices, this paper focuses on malware attacks on a smart inverter. Fig. 1 shows a commercial smart inverter [20]. Specifically, *Network and Application Layer Board (L1)* includes: 1) a microprocessor unit (MPU) with ARM architecture that has relatively high computational power with ROM bootloader and operates smart inverter applications with

This work was supported in part by the National Science Foundation (NSF) under award No. CNS-2219733 and No. CNS-2131163.

978-1-6654-7999-8/22/$31.00 ©2022 IEEE

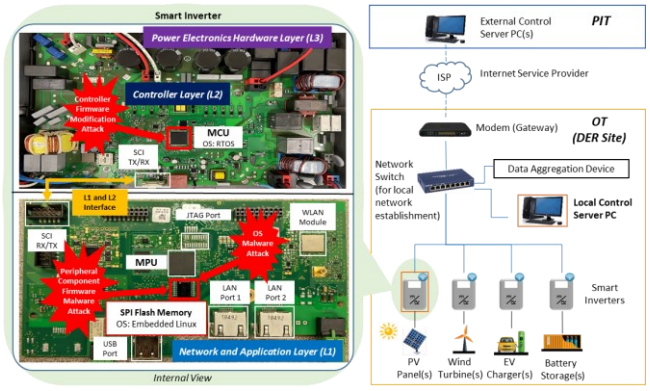

Fig. 1. A commercial smart inverter consisting of three layers: Network and Application Layer (*L1*), Controller Layer (*L2*), and Power Electronics Hardware Layer (*L3*).

Embedded Linux OS; 2) an SPI flash memory that stores bootloaders (e.g., UBoot) and a root file system; and 3) additional peripheral network components and interfaces such as local area network (LAN) ports, a Wi-Fi module, a USB, a serial communication interface (SCI) module, and a JTAG debug port. *L1* is similar to a typical IoT edge devices (i.e., smart grid devices incorporating the network layer) enabling direct connection to external servers in a secure tunnel with TLS 1.2/1.3 via internet. *L1* Linux-based malware files can be loaded to the *L1* using several attack vectors (e.g., remote firmware update and physical reverse engineering).

However, no specific malware security methods were found beyond the firmware and booting security in the smart inverter. The local factory reset could be used if remote patching was disabled by malware. However, a long recovery time is anticipated due to the manual recovery process by physical access. Besides, stolen confidential data from the smart inverter will bring chances for the next ransomware attacks.

In a worst case, a compromised smart inverter or smart grid device by worm might propagate to compromise other connected smart grid devices and the utility control center through the network. The worm-infected utility control center and devices may spread worms to the other power grid stakeholders. If the widespread of the worm attack is successful, there will be severe damages of the power grid and the longest recovery time is anticipated since it is unclear how many devices and systems are infected by worms. Therefore, it is necessary to have a device-specific malware file detection system for the smart grid devices, which can proactively detect the malware files during firmware update and reactively detect malware files from the suspicious smart grid devices.

III. Proposed QCNN-Based Online Malware File Detection

Fig. 2 shows the proposed cloud-based malware file detection pipeline using QCNN with DTL. Once the target file is transferred to the cloud system and then converted into the images using data pre-processing techniques, each datapoint is first passed through a quantum encoding layer to be compatible with the quantum circuit. Quantum Convolution is performed to generate final datapoints after they have been processed by the quantum circuits in the quantum processor. Lastly, they are passed into a fully connected CNN model (i.e., pretrained/basis model) which contains the feature vector layer of the DTL and several dense layers for optimization and normalization. A device specific QCNN model is developed by the retraining the pretrained CNN model (trained by general Linux-based malware and benign files) through the DTL technique with the device specific firmware files. The proposed malware file detection system can be used when the smart grid devices are necessary to update firmware as well as to conduct online malware forensics.

A. Device-Specfic Data Collection and Preprocessing

Benign firmware files can be collected from the Linux file system stored in Serial Peripheral Interface (SPI) flash memory on a target commercial smart inverter [20] and firmware files available from vendors. After a serial connection between the smart inverter and a reverse engineering PC, target firmware files were extracted by exploiting a dump command in flashrom open-source tool. Binary manipulation methods [21] are used to generate variants of Conti ransomware file.

Fig. 3 depicts binary files (a benign firmware file and a malware file) installed in the smart inverter and corresponding binary image files. A static binary firmware file is mapped to an array of integers between 0 and 255. Hence each binary is

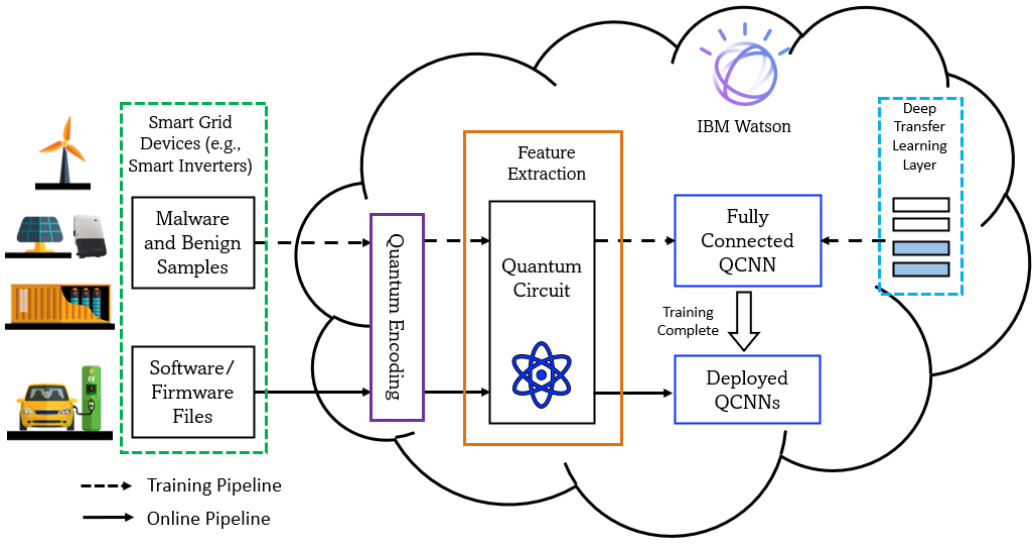

Fig. 2. Proposed cloud-based malware file detection pipeline using IBM Watson and IBM-Q.

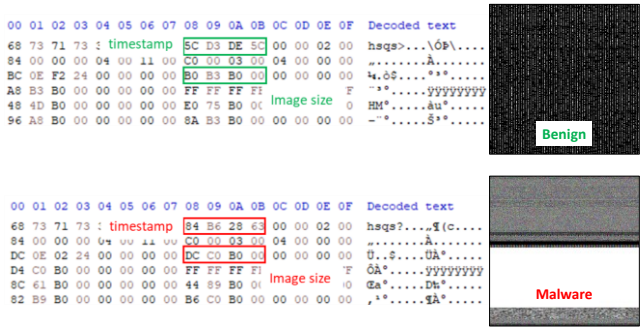

Fig. 3. Binary files and corresponding image files extracted from the smart inverter.

converted into a one-dimensional array [0; 255]. Then the array is normalized to [0, 1] by dividing by 255. The normalized array is then reshaped into a two-dimensional array. The height of the file is the total length of the one-dimensional array divided by the width. Interest readers may refer to [15] for more detail of the data preprocessing.

B. *QCNN Artchiecture*

The proposed QCNN architecture involves encoding pre-processed image data and performing controlled rotations on each datapoint to extract hidden features. Fig. 4 shows a QCNN architecture using Resnet50V2 pretrained CNN model used for malware image file detection. This ensures maximum feature retention and prevents loss of the image file data. The encoding of the classical input values into quantum states is done using the RY gate, which rotates the qubit state around the Y axis. We construct a Sobel-inspired filter using quantum circuits. A series of controlled-rotation gates is used to perform the convolution operation on the quantum state. Using the proposed filter (i.e., Q-filter), better extracted features can be identified rather than a 'random layer' of quantum circuit based on the quanvolutional filter [19]. An example of the Q-filter algorithm is shown in Table I, which converts the image form (128, 128, 3) to (14, 14, 3) when the number of qubits of the quantum hardware is limited to less than seven qubits. To implement a Q-filter in a quantum circuit level, a series of CROT gates are used to perform the convolution operation. The controlled-rotation gates can be implemented using the '`qml.CRot`' PennyLane's quantum gate as shown in Fig. 4. Two CROT gates are coupled together to process each qubit and set the 'alpha' and 'phi' constants to a value of pi/4. This essentially means that the gates will perform a controlled rotation on the target qubit by an angle of pi/4 radians around the x and z axes, respectively.

Table I.
Q-Filter algorithm using Quantum Gates

Input:	*i, j*:Int; *phi*:Intarray
Output:	*phi_updated*:Intarray
1:	**var** phi, i, j:Int; phi_updated:Inarray;
2:	
3:	**function** *Circuit*(phi:Intarray): Intarray;
4:	**for** j **in** range(4)
5:	**qml**.RY(pi*phi[j], **wires** = j) /*encoding data*/
6:	/* Quantum Circuit Using Pennylane*/ **qml**.CRot(**phi** = pi/4, **theta** = 0, **omega** = np.pi/4, wires=[0, 1]) /*operating from q[0] – q[1]*/
7:	**qml**.CRot(**phi** = pi/4, **theta** = 0, **omega** = np.pi/4, wires=[1, 2]) /*operating from q[1] – q[2]*/
8:	**qml**.CRot(**phi** = pi/4, **theta** = 0, **omega** = np.pi/4, wires=[2, 3]) /*operating from q[2] – q[3]*/
9:	**return** phi_updated /* circuit params after quantum evaluation*/

Once the features are mapped, additional Flatten and Dropout layers are added to normalize the datapoints. Finally, the carefully crafted dense layers enable to achieve supremacy when compared to traditional CNN methods for malware image file detection.

C. *Retraining Q-CNN using DTL*

DTL freezes a portion of the fully connected layers, and the last few layers (not frozen) are retrained on the new image datasets. In this paper, the new feature extraction layer is set to be non-trainable. The remaining layers are created as a sequential Keras model with batch normalization, dense layers with dropout and activation functions, and a final output layer with the 'softmax' activation function. we carefully chose hyper-parameters including *loss_functions*, *optimizers* and *learning_rate*. Additionally, optimization involves the use of *Dropout*, *kernel_initializer* and *kernel_regulizer* to prevent overfitting when working with less data counts. Our last effort to reduce overfitting was to use a Leaky_Relu activation function on the last few dense layers and setting the output activation to 'softmax'.

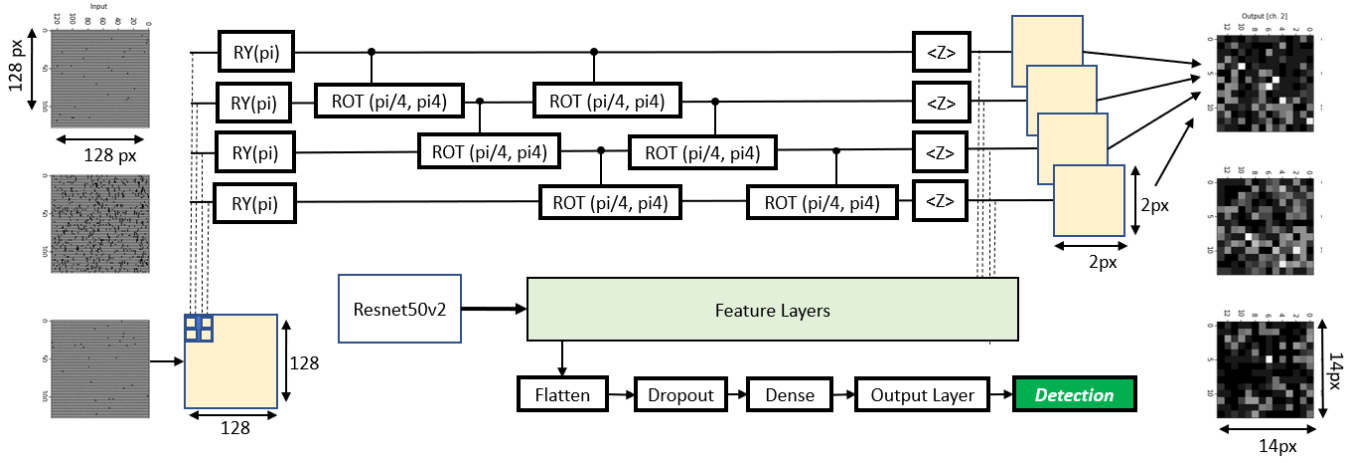

Fig. 4. QCNN architecture using Resnet50V2 pretrained model.

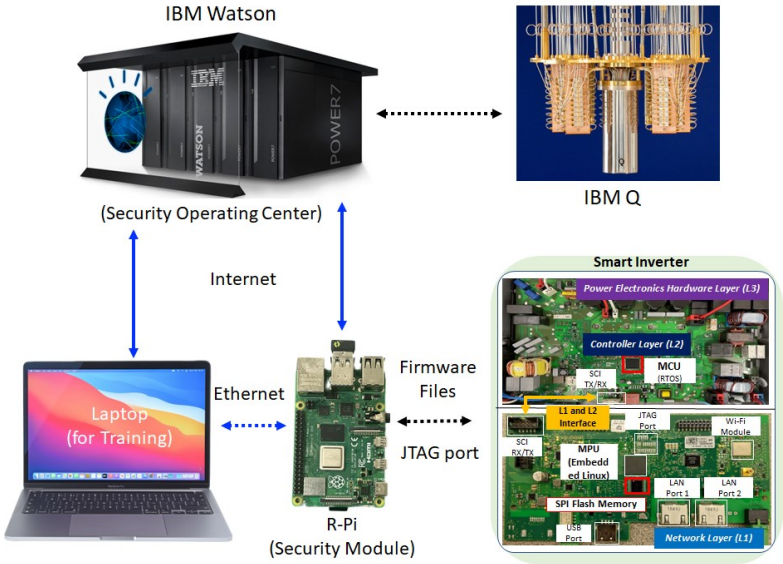

Fig. 5. Experimental setup.

IV. Validation

A. Experimental Setup

Fig. 5 shows the experimental setup to validate the proposed method. The proposed QCNNs with DTL were designed and trained by using the Microsoft Visual Studio Code IDE and Pennylane library [18] (i.e., a python library for QML) in a laptop (Apple M1 SoC, 8 core CPU, 7 core GPU, 8 GBs of RAM). A total of 4 Qubits are used to design the Q-filter and three CNN models (ResNet50V2, Densenet121, and EfficientnetV2) typically used for binary malware image file detection. A total of 1,070 benign programs and 1,130 malware files applicable for an ARM-based architecture and embedded Linux OS are used for training (80%) and validation (20%) using the DTL technique to develop device-specific QCNN models. After that, the deployed models are implemented in IBM Watson cloud platform with 2 core vCPU and 8 GB RAM environment and interfacing the IBM Q quantum processor to perform the online malware file detection using 93 additional testing files consisting of 85 benign and 8 malware files. A raspberry pie (R-Pi) is used as an interface to the smart inverter through the JTAG port and it sends firmware files to the Laptop and the IBM Waston via internet.

B. Results

Fig. 6 shows the comparison of training and validation accuracy curves of the conventional CNN-based malware file detection model (ResNet50V2 with DTL) and the proposed QCNN with DTL based on ResNet50V2. The conventional CNN model requires over 100 epochs to converge in the training stage, while about 40 epochs for training the QCNN model. Therefore, it is evident that using the QCNN using the Q-filter layer significantly improve the detection accuracy.

Table II compares the performance of the four CNN models including the conventional CNN with DTL model and three variants of QCNN with DTL models using the evaluation metrics including Training stage metrics (Training time & Validation Accuracy) using the laptop and deployment stage metrics (Recall, Precision, and F1 Score) using the IBM Watson cloud platform. Overall, QCNN models outperform the conventional CNN model in terms of training time and

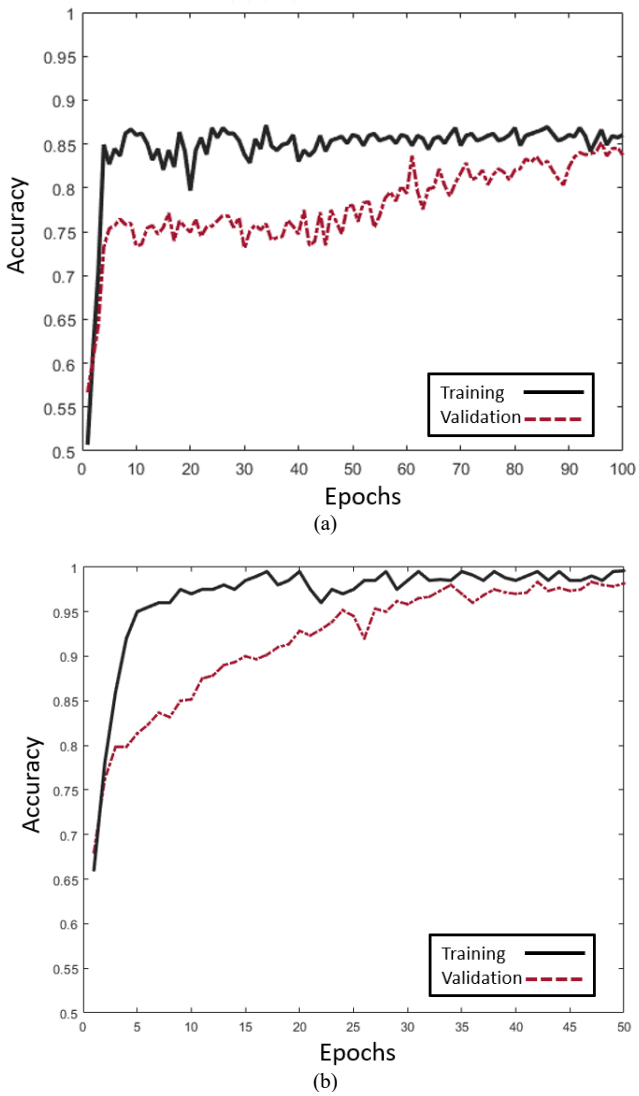

Fig. 6. Accuracy curves of training and validation: (a) classical ResNe50V2 and (b) Q-CNN ResNet50V2.

Table II
Performance Comparison of CNN-based Malware File Detection Models.

CNN-based Models	Training Evaluation		Online Deployment Evaluation		
	Training Time (s)	**Accuracy (%)**	**Recall**	**Precision**	**F1 Score**
Classical ResNet50V2	1484	75.40	0.829	0.867	0.848
Classical VGG-16 [16]	2256	91.31	0.925	0.956	0.942
Q-CNN ResNet50V2	**512**	**97.78**	**0.988**	**1**	**0.994**
Q-CNN DenseNet121	633	95.03	1	0.977	0.951
Q-CNN EfficientNetV2	569	92.31	1	0.966	0.932

accuracy. Overall, QCNN models outperform the classical CNN models due to the enhanced feature extraction by the Q-filter, Among QCNN models, the QCNN coupled with ResNet50V2 achieves the best malware detection rates while requiring the shortest training. In the online deployment experiments using the IBM Watson platform, the proposed QCNN with ResNet50V2 successfully classifies all 8 malware files, with only one misclassification of a benign file as malware. Detection accuracy improvement will be possible by adding more qubits.

V. Conclusion

This paper explored the potential threat of malware attacks targeting a commercial smart inverter and proposed a cloud-based, device-specific online malware file detection method using a cloud-based quantum computing service for smart grid devices. The proposed method utilizes QCNN incorporating quantum circuits with the DTL technique to build a device specific malware file detection. The experimental results show that the proposed method outperforms the classical CNN with DTL methods with higher accuracy with minimum time and effort. The proposed QCNN malware file detection system can be applied to screen new firmware to be installed and to conduct online malware forensics for suspicious smart grid devices. Future works include: 1) investigating more practical malware attack scenarios targeting smart grid devices, 2) studying on malware variant and obfuscation technique to generate potential malware variants, and 3) validating the proposed concept for other smart grid devices.

References

[1] NVIDIA, "Software-defined smart grid meter." [Online]. Available: https://www.enterpriseai.news/2021/12/16/ nvidia-utilidata-partner-on-software-defined-smart-grid-chip-development/

[2] Enphase, "Software-defined micro solar inverter." [Online]. Available: https://enphase.com/installers/microinverters

[3] M. Cokic and I. Seskar, "Software defined network management for dynamic smart grid traffic," *Future Generation Computer Systems*, vol. 96, pp. 270–282, Jul. 2019.

[4] M. Ndiaye, G. P. Hancke, A. B. Abu-Mahfouz, and H. Zhang, "Software-defined power grids: A survey on opportunities and taxonomy for microgrids," *IEEE Access*, vol. 9, pp. 98 973–98 991, Jul. 2021.

[5] P. T. Lee, "The software-defined power grid: How software and sensors are bringing century-old grid technology into the modern age," *IEEE Spectrum*, vol. 5, no. 7, pp. 40–46, Jun. 2020.

[6] B. Li, R. Lu, G. Xiao, T. Li, and K.-K. R. Choo, "Detection of false data injection attacks on smart grids: A resilience-enhanced scheme," *IEEE Trans. Power Systems*, vol. 37, no. 4, pp. 2679-2692, Jul. 2022.

[7] X. Wang, Y. Liu, and K. R. Choo, "Fault-tolerant multisubset aggregation scheme for smart grid," *IEEE Trans. Ind. Informatics*, vol. 17, no. 6, pp. 4065–4072, 2021.

[8] I. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber ware," *Survival*, vol. 53, no. 1, pp. 23–40, Jan. 2011.

[9] D. E. Whitehead, K. Owens, D. Gammel, and J. Smith, "Ukraine cyber-induced outage: Analysis and practical mitigation strategies," in *Proc. IEEE 2017 70th Annual Conference for Protective Relay Engineers*, 2017, pp. 1–8.

[10] U.S. CISA, "Alert (aa20-352a)." [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa20-352a

[11] CNN, "Colonial pipeline ransomware recovered." [Online]. Available: https://www.cnn.com/2021/06/ 07/politics/colonial-pipeline-ransomware-recovered/index.html

[12] Eclypsium, "Conti targets critical firmware," Jun. 2, 2022. [Online]. Available: https://eclypsium.com/2022/06/02/conti-targets-critical-firmware/

[13] T. Z. P. Eder-Neuhauser and J. Fabini, "Malware propagation in smart grid networks: metrics, simulation and comparison of three malware type," *J. Computer Virology and Hacking Techniques*, vol. 15, pp. 109–125, 2019.

[14] A. P. Kuruvila, I. Zografopoulos, K. Basu, and C. Konstaninou, "Hardware-assisted detection of firmware attack in inverter-based cyberphysical microgrids," *Int. J. Electric Power & Energy Systems*, vol. 132, p. 107150, Nov. 2021.

[15] S. Alvee, B. Ahn, T. Kim, Y. Su, Y-W. Yoon, and M-H. Ryu, "Ransomware attack modeling and artificial intelligence-based ransomware detection for digital substations," in *Proc. 2021 6th IEEE Workshop on Electronic Grid (eGrid)*, New Orleans, LA, Nov. 8-10, 2021, pp.1-5.

[16] S. Alvee, B. Ahn, S. Ahmad, K. Kim, T. Kim, and J. Zeng, "Device-centric firmware malware detection for smart inverters using deep transfer learning," in *Proc. 2022 IEEE Design Methodologies Conferences*, Bath U.K., Sep. 1-2, 2022, pp. 1-5.

[17] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, Jan. 2020.

[18] Y. Zhang and Q. Ni, "Recent advancecs in quantum machine learning," *Quantum Engineering*, vol. 2, no. 1, pp. 1-20, Mar. 2020.

[19] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural netowrks," *Nature Physics*, vol. 15, no. 12, pp. 1273-1278, Aug. 2019.

[20] A. M. Jenkins, B. Ahn, A. Akash, and T. Kim, "Device-centric ransomware detection using machine learning-based memory forensics for smart inverters," in *Proc. Eighth Annual Industrial Control System Security (ICSS) Workshop*, Austin, TX, USA, Dec. 6, 2022, pp. 1-7.

[21] A. Abusnaina, *et. al*., "Systemically evaluating the robustness of ML-based IoT malware detectors," in *Proc. 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Taipei, Taiwan, June 21- 24, 2021, pp. 3-4.

[22] A. Mari, "Quanvolutional neural networks — PennyLane," [Online]. Available: https://pennylane.ai/qml/demos/tutorial_quanvolution.html