IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 September 2024, accepted 29 September 2024, date of publication 2 October 2024,
date of current version 14 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3472053

APPLIED RESEARCH

Solar Irradiance Forecasting Using a Hybrid Quantum Neural Network: A Comparison on GPU-Based Workflow Development Platforms

YING-YI HONG[1], (Senior Member, IEEE), DYLAN JOSH DOMINGO LOPEZ[1,2,3], (Member, IEEE), AND YUN-YUAN WANG[4]

[1]Department of Electrical Engineering, Chung Yuan Christian University, Taoyuan 32023, Taiwan
[2]Computer Engineering Department, Technological Institute of the Philippines, Quezon 1109, Philippines
[3]Department of Computer Studies, De La Salle University, Manila 1004, Philippines
[4]NVIDIA AI Technology Center, NVIDIA, Taipei 95050, Taiwan

Corresponding author: Ying-Yi Hong (yyhong@ee.cycu.edu.tw)

This work was supported by the National Science and Technology Council in Taiwan under Grant NSTC 112-2221-E-033-010-MY3 and Grant NSTC 113-2218-E-008-016.

ABSTRACT Modern renewable power operations can be enhanced by integrating deep neural networks, particularly for forecasting solar irradiance. Recent advancements in quantum computing have shown potential improvements in classical deep neural networks. However, current challenges with quantum hardware, such as susceptibility to noise and decoherence, pose risks to its practicality. Hybrid quantum neural networks (HQNNs) are found to mitigate these issues, especially when integrated with graphics processing unit (GPU)-based pipelines. This paper presents a comparative study of different software platforms for developing HQNNs, using multi-location very short-term solar irradiance forecasting as an example. A classical benchmark model is initially designed based on statistical analysis of a 10-minute resolution solar irradiance dataset, with its parameters further optimized using Bayesian Optimization. The experimental design of this paper includes a loss comparison between classical neural networks and HQNNs across different seasons and a performance comparison between Pennylane, Torchquantum, and CUDA Quantum (CUDA-Q) as HQNN development platforms. Experimental results show that HQNNs achieve up to a 92.30% improvement in testing loss compared to classical neural networks. Regarding HQNN development platforms, Pennylane shows an 81.54% testing loss reduction from classical models, Torchquantum shows a 90.34% improvement, and CUDA-Q shows a 92.30% improvement in testing loss. Implementing hardware acceleration libraries for GPU-based state vector simulation demonstrates an approximate 275% speedup in average latency per epoch, a 218% speedup in inference time, and a 10.20% improvement in testing loss compared to CPU-based simulations. CUDA-Q achieves a training time 2.7 times shorter and an inference time 2.9 times shorter compared to Pennylane, while it is 32.3 times faster in training and 31 times faster in inference compared to Torchquantum.

INDEX TERMS Hybrid quantum neural network, solar irradiance forecasting, CUDA-Q, Pennylane, Torchquantum.

NOMENCLATURE

R_z, R_x, R_y	Rotation gates along the $\hat{z}, \hat{x}, \hat{y}$ axes.
ω, θ	Weights of a classical and quantum layer, respectively.
ϕ	Classical data to be quantum encoded.
$U(\phi, \theta)$	A quantum neural network represented as a Unitary.
$\hat{B}_z$	An observable in the z-basis.
$\mathbf{E}_q, \mathbf{E}_q^+, \mathbf{E}_q^-$	Expectation of a quantum layer.
$J_\theta, J_{\theta+\frac{\pi}{2}}, J_{\theta-\frac{\pi}{2}}$	A transformation considering a quantum. layer and an observable.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaodong Liang.

∇_q, ∇_c	Gradient of a quantum and classical layer, respectively.
x, y	Input and output vectors for a neural network model.
b	The batch size of an input vector.
Λ	Number of locations in an input vector.
Φ	Number of features in an input vector.
T	Planning horizon period in an input vector.
T_{season}	Total time in a season.
X, Y_{gt}	Dataset used for training given the features and ground truths, respectively.
z	Activation of each classical neural network layer.
κ	Number of variables for a combination of parameters.
h	A convolution kernel.
$l, \Delta l$	Lag and lag increment.
k_1, k_2	Number of kernels of convolution layers.
c	Number of channels of convolution layers.
u	Number of neurons of a dense layer.
p, p'	Configuration sample of parameters of a model.
p^*	Optimal parameter configuration.
$\mathbf{P}$	Search space of all combinations of parameters for a model.
$\mathcal{P}$	Set of feasible configurations with corresponding model loss.
ε	Threshold constant for the stopping criterion.
σ	Gaussian noise standard deviation.
f_z, f_s	Zero measurement and spiking measurement frequency.
s	Model sensitivity.
r()	A rectified linear unit (ReLU) function.
F()	A flatten function.
$a()$	Acquisition function of the Bayesian Optimization Algorithm.
$s()$	Stopping criteria function of the Bayesian Optimization Algorithm.
$\mathbf{m}()$	The objective function pertaining to the hybrid quantum model.
$U_{R_{yxz}}(\phi)$	Unitary for encoding classical data.
$U_{R_{xyz}}(\theta)$	Trainable parametric quantum circuit.
$\hat{X}$	A CNOT gate.
U_c	Unitary representing circular entanglement.
$U_q(\theta)$	Unitary for a single parametric quantum neural network block.

I. INTRODUCTION

The operation of power generation from renewable energy sources is essential for modern power systems. However, it presents a challenge for power grid operators due to the inherent uncertainty of sources such as wind and solar energy [1]. Modern energy research employs artificial intelligence, specifically deep learning models, to profile and predict renewable power generation with a higher level of certainty [2], [3], [4], [5].

A recent trend in employing deep learning models for predicting renewable energy generation involves quantum computing [6], [7], [8], [9], [10]. Quantum computing is an emerging technology that leverages concepts such as quantum bits or "qubits," superposition, and entanglement to represent and process information [11], [12], [13]. These concepts are not inherent in existing computing paradigms, referred to as "classical." Recent developments in quantum computing have demonstrated its applicability to machine learning tasks, including classification [14], [15], [16] and regression [17], [18], [19]. However, quantum computing in industrial practice faces challenges due to current quantum hardware limitations, such as the limited number of qubits available. This limitation can result in insufficient data for mitigating errors in reading out information from qubits [20], [21], and it may not be suitable for handling high-dimensional data or the weights within a neural network [22], [23].

A new paradigm in machine learning and quantum computing has been introduced to address the challenge of insufficient qubits for neural networks: hybrid quantum neural networks (HQNNs). HQNNs are variants of variational quantum circuits in which central processing units (CPUs) or graphics processing units (GPUs) are used to optimize quantum circuits alongside classical neural networks. In recent years, efforts have been made to employ HQNNs for renewable energy prediction. For instance, wind energy prediction using quantum neural networks (QNNs) has been explored by several researchers [7], [8], while solar irradiance forecasting has been achieved by [6], [9], and [24].

The training and deployment of large neural networks often involve high-performance computing (HPC) setups and typically utilize hardware acceleration libraries such as CUDA. A similar trend is observed for HQNNs, where training these models can benefit from the same hardware acceleration used for classical neural networks. As HQNNs mature and gain traction in academia and industry, several libraries and platforms have emerged to support their development. Platforms such as Pennylane [29] are widely used across various disciplines, including power engineering and renewable energy planning. When selecting libraries for HQNN development, considerations include model performance, ease of use, documentation and support, and applicability for deployed systems with limited resources. Although hybrid quantum deep learning approaches for renewable energy forecasting already exist, this paper utilizes a deeper parametric quantum circuit (PQC) than those in [6], [9], and [24], resulting in improved regularization and reduced testing loss for smaller classical neural network architectures. Additionally, this paper explores various hybrid quantum computing platforms beyond Pennylane, including CUDA

Quantum (CUDA-Q) and Torchquantum, to demonstrate their applicability in operationalizing quantum computing for AI in energy operations and planning.

This paper presents a comparative analysis of Global Horizon Irradiance (GHI) prediction using three existing hybrid quantum deep learning platforms, focusing on HPC and cost-effective GPU workstations. The contributions of this paper include:

- This paper proposes a statistics-driven architecture design for classical neural networks, specifically tailored for northwestern Taiwan, considering cross-sectional and solar GHI time-series characteristics of the data. By utilizing a Bayesian Optimization Algorithm, the parameters of the classical neural network are optimized to minimize validation loss.
- This paper presents a performance analysis comparing classical and HQNNs using a multi-location very short-term solar irradiance forecasting model across multiple deep learning platforms.
- This paper presents a comparative analysis of different HQNN development platforms, considering their loss and timing performances. This analysis aims to determine the advantages and disadvantages of using specific platforms for the development HQNNs.

The paper is structured as follows: Section II discusses an overview of the related works. Section III discusses the basic concepts of quantum computing and information representation, neural networks, and different implications in the industry. Section IV introduces three hybrid quantum deep learning platforms and provides a comparison. Section V outlines the methodology adopted in this study, including the model architectures and parameter optimization. Section VI presents the experimental results. Finally, Section VII concludes the paper and discusses future works.

II. RELATED WORKS

Solar irradiance forecasting is crucial for optimizing renewable energy utilization. Numerous methods have been employed to enhance the accuracy of solar irradiance forecasting models. Early time-series forecasting approaches include statistical models such as ARIMA, SARIMA, and ARIMAX, which rely on autoregressive and moving average statistics [25], [26]. While these models are effective for small datasets, they become less accurate when applied to larger datasets with longer time horizons, spanning decades, or high-resolution data with minute, second, or millisecond sampling rates, where they fail to generalize effectively over extended planning periods.

A. CLASSICAL DEEP LEARNING APPROACHES

Recent advances in solar irradiance forecasting leverage deep learning techniques, including multi-layer artificial neural networks (ANNs), convolutional neural networks (CNNs), and recurrent networks such as long short-term memory (LSTM) networks [4], [27]. Many contemporary deep learning-based solar irradiance forecasters use hybrid or ensemble models. A common approach combines CNNs and LSTMs, where CNN layers extract high-dimensional spatial features and LSTM networks capture temporal dynamics [2], [28]. However, LSTM-based models can suffer from the vanishing gradient problem, especially in deeper structures or with longer memory sequences. This issue is often addressed using attention mechanisms, as seen in attention-based models like transformers. Sharda et al. [29] employed a self-attention model for multi-horizon solar irradiance forecasting using the National Renewable Energy Laboratory (NREL) benchmark dataset, which improved learning over intermittency and provided more robust predictions. Similarly, Zhang et al. [30] enhanced the robustness of solar irradiance forecasters against incomplete data with a hybrid transformer network.

B. HYBRID QUANTUM DEEP LEARNING APPROACHES

Quantum computing has the potential to enhance machine learning and deep learning models for specific problems. Emmanoulopoulos and Dimoska [31] demonstrated that time-series forecasting can be improved by incorporating quantum computing-based layers into neural networks. These approaches modify existing deep learning architectures by adding or substituting quantum computing layers. Evidence of quantum computing's application in renewable energy forecasting is already apparent. For instance, hybridized neural networks for wind energy forecasting have utilized variational quantum eigensolvers (VQEs) combined with LSTM architectures [7], [8], with Pennylane and the lightning-qubit configuration [32]. Solar irradiance forecasting has also benefited from hybrid quantum neural networks (HQNNs), as seen in [6], where a variational quantum circuit (VQC) was integrated into an LSTM gate rather than as a neural network header. Sagingalieva et al. [9] introduced several hybrid quantum architectures, including HQNNs, hybrid quantum LSTMs (HQLSTMs), and hybrid quantum sequence-to-sequence (HQSeq2Seq) networks, showing that HQLSTMs outperform other models in accuracy and model loss when using Mediterranean PV data [33]. Similarly, Sushmit and Mahbubul [24] developed a variant of HQNNs by incorporating feed-forward ANNs with quantum neural network (QNN) layers, which demonstrated promising results in solar irradiance forecasting compared to bidirectional LSTMs. Like their counterparts in wind speed forecasting, the studies in [9] and [24] employed Pennylane for developing their hybrid quantum models.

This paper presents an approach for forecasting solar irradiance using HQNNs applied to the Taoyuan Region of Taiwan. Taiwan's distinct four seasons, each with significant variations in means and standard deviations, necessitate different architectural adaptations to effectively capture localized seasonal trends. The study will also incorporate robust tests similar to those in [29] and [30] to evaluate HQNNs' performance under noisy and extreme conditions. Additionally, it reviews various platforms for developing HQNNs, with a focus on ensuring GPU acceleration across all

TABLE 1. Gap analysis of related works in solar irradiance forecasting using hybrid quantum neural networks.

	Planning Horizon	Dataset used	Hybrid Quantum Models Used/Compared	Hybrid Quantum Platform Used	Parameter Optimization	Operational Capabilities Measured
Yu et al. [6]	1 hour-ahead	https://nsrdb.nrel.gov/ (China)	HQLSTM	Pennylane	Preset	MAE, RMSE, R^2
Saginaglieva et al. [9]	1 hour-ahead	Malvoni, De Giorgi, and Congedo [33]	HQNN, HQLSTM, HQSeq2Seq	Pennylane (lightning-qubit)	Optuna optimizer	MAE, RMSE, R^2
Sushmit and Mahbubul [24]	1 hour-ahead	NASA POWER (Bangladesh)	QNN, HQNN	Pennylane (lightning-qubit)	Preset	MAE, RMSE
This work	10, 20, …, 60 minute-ahead	https://nsrdb.nrel.gov/ (Taiwan)	HQNN, HQCNN	Pennylane, Torchquantum, CUDA-Q	Bayesian Optimization	RMSE, Inference Speed, Training Time

HQNN layers—an aspect not commonly addressed in prior literature, where lightning-qubit configurations typically rely on CPU-based forward and backward propagation. A gap analysis comparing the works in [6], [9], and [24] with the proposed method for solar irradiance forecasting using HQNNs is provided in Table 1.

III. QUANTUM COMPUTING AND QUANTUM MACHINE LEARNING CONCEPTS

A. QUANTUM INFORMATION AND COMPUTING

In quantum computing, information processing occurs through encoding available data onto quantum bits or qubits. Qubits are mathematically represented as vectors and carry this information within a quantum computer. The advantage of having qubits over classical bits is that they can enjoy quantum properties such as superposition and entanglement.

Qubits can exist in a superposition of states, enabling them to represent more information than classical bits. While classical bits can only hold a value of 0 or 1 at any given time, based on their energy potential, qubits can represent both values simultaneously in a superposition. The energy potential of a qubit is linked to its amplitude, and the probability of measuring a particular state of the qubit is given by the squared root of the amplitude. According to Born's rule, the sum of all measurement probabilities must equal 1. For example, if a qubit has an amplitude of $1/\sqrt{2}$ for both $|0\rangle$ and $|1\rangle$, the probabilities of measuring $|0\rangle$ and $|1\rangle$ are both 50%, as illustrated in Fig. 1.

Multiple bits combine to form a binary word, which is organized into a register. Similarly, multiple qubits can form a quantum register. The size of a classical register is directly equivalent to the number of bits it contains; thus, a classical register of size n has n classical bits. In contrast, a quantum register, consisting of n qubits, can represent 2^n different states simultaneously due to superposition. Therefore, the size of a quantum register is 2^n, reflecting its ability to encode a vast amount of information compared to its classical counterpart.

Operations on qubits are performed using specialized functions known as quantum gates, represented by Hermitian matrices. These quantum gates act as linear operators tailored for manipulating prepared qubits. These quantum gates can be put into an ensemble of operations to form a quantum. A special type of quantum gate that is parametric is called a Rotation Gates where specific phases can be input to induce vector rotations about either in the $\hat{x}$, $\hat{y}$, or $\hat{z}$-axis, as seen in Fig. 2.

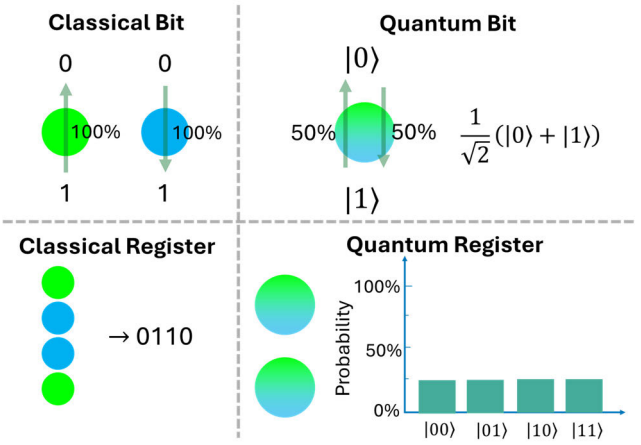

FIGURE 1. Classical and quantum information representation.

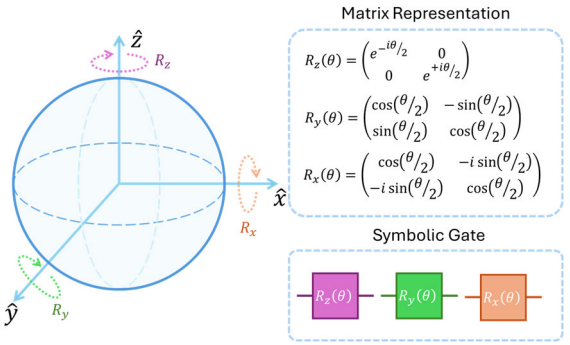

FIGURE 2. Representations of the rotation gates Rx, Ry, and Rz.

B. QUANTUM NEURAL NETWORKS

When different rotation gates, such as R_z, R_y, R_x, or generic Unitary gates, are compounded in a quantum circuit and may form a Parametric Quantum Circuit (PQC) [34]. PQCs

may be used as variational quantum circuits so that the angles in each rotation gate can be fine-tuned to modify the measurement of a quantum circuit. This idea led to using PQCs as an analog to a multilayer Perceptron, which can then be called a Quantum Neural Network (QNN).

1) QNN ARCHITECTURE

As depicted in Fig. 3, a basic QNN consists of three main components: an encoder, an ansatz, and a measurement part. The encoder transforms classical inputs or activations from previous neural network layers into a format understandable by a quantum circuit. Various methods for encoding classical data onto a QNN include Angle or Phase Embedding, Amplitude Embedding, Instantaneous Quantum Polynomial (IQP) Embeddings, and other custom embedding circuits.

The next component of the QNN is the parametric ansatz. The ansatz adjusts parameters to transform the input into the desired output. These parameters, serving as the weights of the QNN within the deep learning model, typically require rotation gates or unitary gates to receive updatable parameters. In cases where QNN outputs exceed one qubit (for binary classification), entanglement between qubits is applied after the rotation gates. Entanglement layers enforce weight-tying between rotation gates, akin to weight-tying of neurons in classical neural networks.

The final component of the QNN is the measurement layer, where measurement gates are applied to observe the quantum circuit, usually in the Z-basis. Measuring the quantum circuit enables the transfer of processed quantum information from qubits to classical bits.

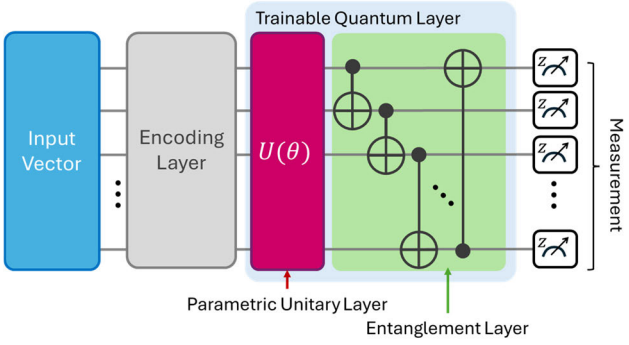

FIGURE 3. Simplified quantum circuit for a single trainable quantum neural network layer.

2) PARAMETER SHIFT RULE

Another consideration in designing and implementing HQNNs is coordinating the weight update routine of each layer. Computing the gradient of a classical layer with a quantum layer cannot be done in the same fashion as classical neural networks. Weight updates in the quantum layer use the Parameter Shift Rule [35] to compute its gradient instead. The expectation of the quantum layer $\mathbf{E}_q$ is computed with respect to the weights of the previous layer ω and the parametric phases of the layer θ as seen in (1).

$$\mathbf{E}_q(\omega;\theta) = \langle\omega| J_\theta\left(\hat{B}_z\right)|\omega\rangle = \langle\omega| U^\dagger(\theta)\hat{B}_z U(\theta)|\omega\rangle \quad (1)$$

A gradient depends on the measurement of the quantum layer $U(\theta)$ with a corresponding observable $\hat{B}_z$ wherein the z-basis is used in this paper. A shift factor is needed to apply the parameter shift rule to compute the gradient for a quantum layer. In this implementation, the shift factor is supposed to be $\frac{\pi}{2}$. The parameter-shift rule-based gradient is presented in (2), where ∇_q represents the gradient of the quantum neural network. This gradient is computed as the difference of half of each expectation, with angles shifted by a factor of $\pm\pi/2$. Finally, if the QNN is preceded by a classical neural network layer, the quantum gradient is computed with the gradient of the preceding layer, as seen in (3).

$$\nabla_q(\omega;\theta) = \frac{1}{2}\begin{pmatrix}\left\langle\omega\left|J_{\theta+\frac{\pi}{2}}\left(\hat{B}_z\right)\right|\omega\right\rangle - \\ \left\langle\omega\left|J_{\theta-\frac{\pi}{2}}\left(\hat{B}_z\right)\right|\omega\right\rangle\end{pmatrix} \quad (2)$$

$$\nabla_q(\omega;\theta) = \frac{1}{2}\left(\mathbf{E}_q^+ - \mathbf{E}_q^-\right)\nabla_c \quad (3)$$

C. INDUSTRIAL APPLICATION CHALLENGES OF QUANTUM DEEP LEARNING

Although recent years have demonstrated proof-of-concept for the technical applicability of quantum computing, the adoption of quantum deep learning in industrial settings still faces several challenges

1) SCALABILITY

The effectiveness of deep learning models in feature extraction is closely linked to the depth and dimensionality of their architectures. While quantum computing offers substantial technical advantages, the current limitation in the number of available qubits prevents the creation of purely quantum counterparts to existing state-of-the-art models. To address this, the current approach involves combining classical neural networks with a limited number of quantum neural network layers that can be represented by the available qubits. This integration results in HQNNs, leveraging the strengths of both classical and quantum systems for enhanced feature extraction and information representation.

2) AVAILABILITY

Currently, most resources for running quantum circuits are accessible only through cloud platforms, which can hinder the development and validation of quantum circuits and HQNNs. The absence of local execution capabilities may delay progress in engineering these systems. To address this, simulations can be run on local computers using central processing units (CPUs), but this approach presents challenges for neural network training, as best practices recommend using graphics processing units (GPUs). Training bottlenecks arise when feature vectors processed by multiple GPUs are retrieved and handled by a single CPU, affecting both feed-forward and gradient-update operations.

Recent advancements in quantum circuit simulation leverage GPUs for quantum operations, as quantum information is represented as vectors and matrices that GPUs can efficiently process. This paper demonstrates the use of GPU resources for both classical and quantum layers in training an HQNN model, showcasing the benefits of GPU acceleration in this context.

3) DEPLOYMENT

AI's widespread adoption across various industries is driven by its versatility in deployment across different platforms, including embedded systems, mobile phones, and the cloud. Another key factor contributing to AI's success in intelligent systems is its operational speed. While quantum circuits can be accessed through cloud services, deploying them as saved models may require additional effort. The operational speed of quantum circuits is largely dependent on the quality of the hardware on which they are executed.

4) ENVIRONMENTAL AND ETHICAL IMPLICATIONS

Although quantum computing is still in its early stages, its potential implications should be proactively considered to avoid regulatory issues and mishandling, similar to concerns associated with AI. One significant challenge facing quantum computing, particularly superconducting qubits, is high electricity consumption, which hampers the scalability of these systems. To address this, research is underway into alternatives such as neutral atom and photonic-based quantum systems, which promise reduced electricity consumption and a smaller carbon footprint. Quantum computing holds the potential for substantial advancements in various fields within computer science and engineering, potentially leading to the hyper-automation of processes. While hyper-automation can offer significant business benefits, it is crucial to establish regulations to ensure that safety-critical processes and the handling of personal and private information remain secure and well-regulated.

IV. HYBRID QUANTUM DEEP LEARNING DEVELOPMENT WORKFLOW

Designing HQNNs closely resembles designing and developing classical neural networks. Fig. 4 illustrates a generic HQNN development workflow, considering the diverse usage of CPUs and GPUs for data handling, model development, and quantum circuit simulation tasks. Hybrid quantum development platforms commonly used are also introduced in this section, including their simulation capabilities on GPU hardware and their compatibility with hardware accelerators.

A. HQNN DESIGN AND DEVELOPMENT

Data preparation tasks such as data wrangling, cleaning, statistical analysis, aggregation, and splicing are typically accomplished using CPU-based computing. The design of HQNNs involves both classical deep neural network model architecture programming and quantum circuit encoding as quantum neural network layers. The development of classical and quantum layers can be determined by whether they can be loaded onto the GPU for access to GPU-based acceleration. The architecture model and prepared dataset may also be loaded to the GPU to prepare for optimization. Optimization of HQNNs is variational since it would involve an external optimization algorithm to tune the weights of a quantum layer. Feedforward and backward propagation consider both classical and quantum layers as part of the total loss of the HQNN model. Operations on the classical layers can improve performance using specific hardware acceleration libraries such as cuDNN [36], and quantum simulations would benefit from cuQuantum [37]. However, backpropagation considering the quantum layer would use the parameter shift rule to compute its gradient.

B. DEVELOPMENT PLATFORMS

The implementation and development of hybrid quantum deep learning models require specialized software. In recent years, Python has become one of the preferred programming languages for implementing quantum computing, aside from C++ and Rust [38]. Software packages such as Qiskit [39], Cirq [40], and Q# [41] are also popular among quantum machine learning practitioners and researchers. While Qiskit has been widely utilized in quantum machine learning research and TensorFlow Quantum [42] in production settings, this paper focuses on libraries or platforms tailored for hybrid quantum deep learning. These platforms are designed to leverage GPU acceleration and are compatible with similar classical deep learning backends. Furthermore, this subsection discusses three platforms: Pennylane, CUDA Quantum, and Torchquantum. A high-level comparison is presented in Table 2.

1) PENNYLANE

Pennylane [29] is a Python-based platform developed by Xanadu, Inc., focusing on variational quantum circuits. The platform adopts an object-oriented Pythonic approach to building and training quantum circuits. With Pennylane, quantum devices can be configured to run on a local device's CPU, GPU, or even on an external quantum computer.

Several initiatives in hybrid quantum computing for renewable energy forecasting [7], [8] have predominantly utilized Pennylane due to its compatibility with other deep learning platforms like Keras and PyTorch. This cross-compatibility facilitates easy integration with existing neural networks, enabling the addition or modification of classical layers as QNNs.

In Pennylane, GPU acceleration is achieved through the 'lightning-gpu' device configuration, which leverages cuQuantum, specifically cuStateVec within cuQuantum. Developed by NVIDIA, Inc., cuQuantum is a comprehensive software development kit (SDK) comprising low-level primitives, including cuStateVec and cuTensorNet. cuStateVec focuses on state vector simulation of quantum

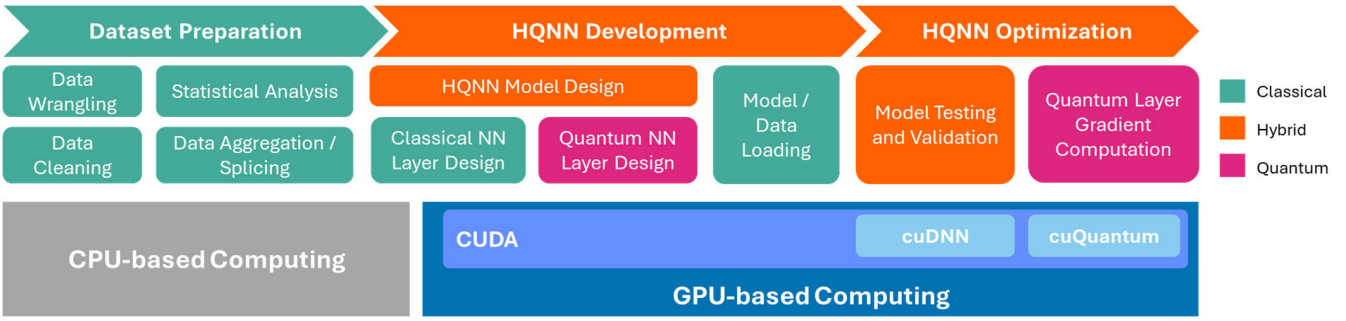

FIGURE 4. A generic GPU-based workflow for Hybrid Quantum Neural Network development.

TABLE 2. Summarized high-level comparison of Pennylane, Torchquantum and CUDA-Q.

Hybrid Quantum Development Platforms			
Considerations	**Pennylane**	**Torchquantum**	**CUDA-Q**
QNN layer programming	Functional	Functional	Explicit
Quantum simulation backends	CPU-based simulator, GPU-accelerated, cuQuantum	CUDA Only, cuQuantum (not implicitly integrated)	CPU-based simulator, GPU-accelerated, cuQuantum (supports asynchronous parallel simulation)
Implementation of parameter shift rule	Built-in but modifiable	Explicit	Explicit
Interoperability with various quantum hardware	Yes	Currently, only IBM Qiskit	Yes
Documentation	Rich documentation for Hybrid QNNs with tutorials	Existing documentation on HQNNs with tutorials	Existing documentation on quantum circuits with tutorials

circuits, offering significant speedup and efficient memory usage. Conversely, cuTensorNet accelerates tensor-network-based simulations, representing quantum circuits as tensor networks. Pennylane with cuQuantum acceleration demonstrates a notable speed advantage over the 'lightning-qubit' configuration for operations requiring a large number of qubits.

2) CUDA QUANTUM

CUDA Quantum [43], or CUDA-Q for short, is an open-source quantum computing programming platform launched by NVIDIA, Inc. The platform focuses on interoperability with existing quantum hardware, GPUs, and CPUs in a single system. The CUDA-Q toolkit offers C++ and Python versions for tightly coupled hybrid quantum-classical system developments. Since CUDA-Q is relatively new, specific methods or sub-packages dedicated to HQNNs have yet to be developed. However, as of this writing, the open-source software and official documentation provide examples of HQNN training. One advantage of the platform is the ability to parallelize quantum workstreams across multiple GPUs, HPC nodes, or multiple available quantum processing units (QPUs).

3) TORCHQUANTUM

In 2022, Wang et al. from MIT developed Torchquantum [44], a library specifically designed for the Torch deep learning platform to facilitate the implementation of HQNNs within Torch architectures. This library leverages CUDA acceleration, which is the native accelerator for PyTorch deep learning pipelines. While it is possible to convert and accelerate quantum circuits using cuQuantum, this functionality has not yet been integrated as a built-in feature within the package.

V. METHODOLOGY

Comparing different HQNNs necessitates a standard comparison between classical neural network architectures and hybrid quantum neural network architectures. This paper presents a study for solar irradiance prediction to illustrate these comparisons in the context of renewable energy operations.

A. DATASET PREPARATION

Classical and hybrid quantum models would use a solar irradiance dataset from the National Solar Radiation Database (NSRDB) [45] provided by NREL. The dataset contains spatial and temporal features and locations in available meteorological data points of the United States and a subset of international locations. Multiple features such as Global Horizon Irradiance (GHI), Diffused Horizontal Irradiance (DHI), Diffused Normal Irradiance (DNI), wind speed, wind direction, surface albedo, ambient temperature, precipitation, solar zenith angle, cloud types as well as clear sky GHI, DHI, and DHI are considered. The dataset consists of 2,893,968 samples collected at 10-minute intervals from 2016 to 2021, spanning 11 locations across Taiwan (see Fig. 5). Data from 2021 is designated as the test set, while the remaining samples

are used for training and validation. However, 10-minute resolution data for the years 2022-2023 were unavailable in the source database at the time of this study.

The input vector x for the neural network is prepared as a three-dimensional vector so that multiple predictors may be considered compactly. The input then has a shape of $(\Lambda, \mathrm{T}, \Phi)$ wherein the symbol Λ represents the number of locations; the shifted time sample T means a time horizon of 60 minutes for different seasonal periods T_{season}; and Φ denotes the number of multiple features. Supporting statistics for identifying Φ and T are shown in the subsequent subsections. The output vector y is a one-dimensional vector (GHI) with a size of 6 (i.e., 1 hour-ahead; 6 × 10 minutes), as illustrated in Fig. 5.

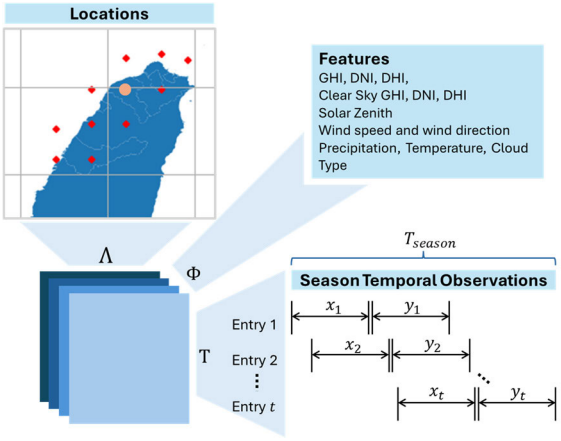

FIGURE 5. Characterization of the solar irradiance forecasting model's input/output vector.

B. STATISTICS-DRIVEN ARCHITECTURE DESIGN FOR LOCALIZED SOLAR IRRADIANCE FORECASTING

Developing a generic set of initial spatial and temporal features is challenging due to the variability and locality of meteorological data, which are influenced by a country's climate, topology, and the presence of microclimates. This study presents a localized analysis of northwestern Taiwan, focusing on a GHI-centric statistical approach to inform the design specifications of the initial classical deep learning model architecture. The statistics-driven design includes correlation analysis for cross-sectional features, time-series analysis for temporal features, dimensionality reduction, and further architecture optimization using Bayesian Optimization.

1) CORRELATION ANALYSIS

The features used for modeling the solar irradiance forecasting model were initially filtered using statistical methods. Cross-sectional analysis among the features was conducted using Pearson correlation to identify linear relationships and Spearman correlation to identify non-linear relationships. As shown in Table 3, the correlation analyses reveal that the features Surface Albedo, Latitude, and Longitude have the least correlation with the target feature GHI. Additionally, the selected locations (10 red dots in Fig. 5) were subjected to correlation analysis to determine their significance to the target location (Taoyuan City in Taiwan (24°59'29''N, 121°18'52''E), orange dot in Fig. 5). The Pearson and Spearman correlation statistics indicate that all other locations have considerable correlation with the target location. The photovoltaic (PV) capacity studied in Taoyuan City, Taiwan, is 55.324 MW, while the total installed PV capacity across Taiwan is 13.517 GW, with a system peak load of 41.42 GW in 2024. Accurate solar irradiance forecasting is crucial, as PV power generation impacts the scheduling of gas-turbine generators and pumped-storage units, which are used to compensate for the intermittent power generation from PV farms.

2) TIME-SERIES ANALYSIS

Individual features of the dataset are subjected to time-series statistical analysis due to their temporal nature. The goal of the time-series analysis is to (1) identify the ideal lag of the GHI and (2) remove features and locations (red dots in Fig. 5) that do not contribute to the target (Taoyuan City, represented by the orange dot in Fig. 5) in the temporal aspect. The samples are temporally divided by season, as Taiwan experiences four distinct seasons: winter, spring, summer, and autumn. Fig. 6 (a) illustrates the monthly GHI distribution, while Fig. 6 (b) presents the seasonal distribution. The trends show significant variability across the seasons, prompting this study to propose four distinct forecasting models, each tailored to a specific season.

a: GHI LAG

Autocorrelation and partial autocorrelation statistics indicate that a lag (l) of 6 is most significant. Therefore, the value of t in the input data is set to 6. Additional stationarity tests were conducted on the target feature to assess its viability for time-series prediction.

b: TEMPORAL FEATURE PRUNING

Granger causality tests were performed for each pair of features and locations to determine their viability as time-series predictors for the GHI of the target location. The results, shown in Table 3, indicate that all features, except for Surface Albedo, exhibit causality with respect to GHI. Consequently, Surface Albedo is excluded, while all locations are retained due to their demonstrated causality. Thus, the Λ dimension of the model input vector is set to 11, ensuring that all locations are considered significant in relation to the target location.

The number of features to be used, Φ, is set to 12. This is because a total of three features were removed: longitude and latitude were eliminated based on the cross-sectional analysis, and Surface Albedo was removed based on the Granger causality test.

3) DIMENSION REDUCTION

Principal Component Analysis (PCA) is a widely used unsupervised machine learning technique for dimensionality reduction. In this study, PCA was applied to the filtered

TABLE 3. Correlation and Granger causality for feature analysis.

Statistic	Features						
	DHI	*DNI*	*Surface Albedo*	*Wind Direction*	*Wind Speed*	*Clear Sky DHI*	*Clear Sky DNI*
Pearson (statistic)	0.969	0.897	0.029	0.193	0.042	0.919	0.966
Spearman (statistic)	0.8	0.873	0.022	0.26	-0.044	0.666	0.839
Granger Causality (p-value)	0.00	0.00	**0.379**	0.000	0.000	0.000	0.000
	Clear Sky GHI	*Solar Zenith Angle*	*Temperature*	*Precipitable Water*	*Cloud Type*	*Longitude*	*Latitude*
Pearson (statistic)	0.966	-0.9	0.324	0.085	-0.131	-0.025	-0.023
Spearman (statistic)	0.863	-0.76	0.352	0.083	-0.193	-0.046	-0.042
Granger Causality (p-value)	0.00	0.00	0.000	0.000	0.000	**0.613**	**0.557**

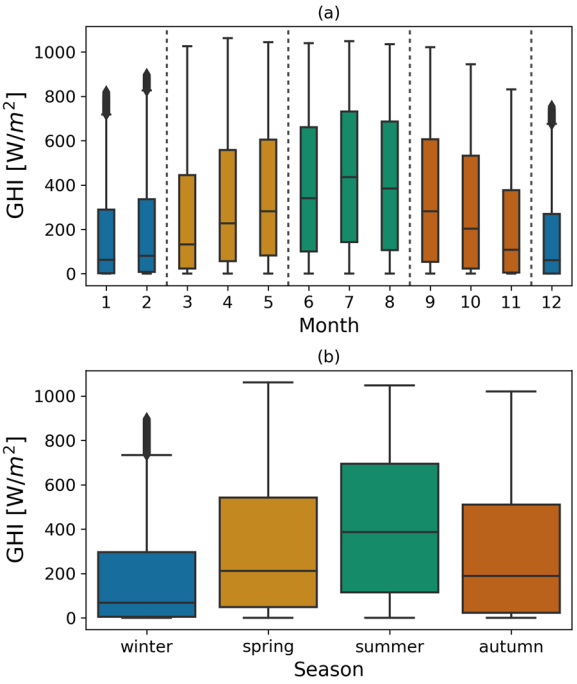

FIGURE 6. GHI distribution in Taiwan in (a) monthly resolution and in (b) seasonal resolution.

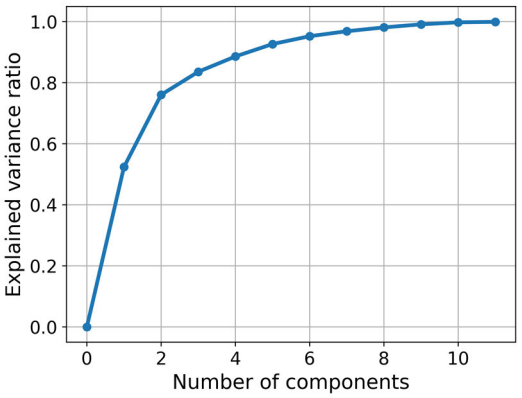

FIGURE 7. Characterization of the solar irradiance forecasting model's input/output vector.

features alongside traditional statistical analysis. Fig. 7 illustrates the relationship between the reduced feature dimensions and their explained variance. Following Occam's razor, the optimal representation of the feature set is achieved with approximately three components. A separate dataset was prepared using 12 features and a 3-component representation, which will be used to train the deep learning models.

C. CLASSICAL CONVOLUTIONAL NEURAL NETWORK (CNN)

The architecture of the baseline model for comparing the HQNN is described in this section. A convolutional neural network (CNN) is used as the feature extractor because the input vector for the solar irradiance forecasting model is a three-dimensional matrix with a batch size of b.

1) NEURAL NETWORK DESIGN

The classical neural network in Fig. 5 (a) is comprised of two convolution layers implemented as (4) whereas $z^{(\ell)}_{Conv}(c, k_\ell)$ represents generic convolution layer in the ℓth order of a neural network. It is noted that the input vector x has a lag of l and an additional variable delay of Δl depending on the season. The general lag l is the baseline lag for the entire dataset. A variable lag Δl is introduced as part of optimizable parameters p to capture different periodicity within seasons. The convolution operation, $\circledast$, involves the input vector from the previous layer and a kernel h_ℓ, which has a shape that varies on dimensions k_ℓ and c. The output tensor of the convolution is then subjected to a ReLU activation function represented as r(). The convolutions are succeeded by a flatten layer in (5) whereas F is a flattening function that transforms $z^{(2)}_{Conv}$ that has a shape of (H, W) into a one-dimension vector of size H $\times$ M. The flatted feature vector z_{flat} is then inputted to fully connected neural networks or dense layers as seen in Fig. 8. The dense layers mathematically represented in (6) and (7) are functions of a variable neuron count u. The shape of the weight ω_1 is defined by the shape of z_{flat}, which is denoted as s_F and u. Finally, the output layer in (7) produces the weight ω_2 that has a shape of (u, s_y), where s_y is the expected size of the prediction $\hat{y}$ which is 6.

$$z^{(1)}_{Conv}(\Delta l, c, k_1) = \mathrm{r}\left(\begin{matrix}\mathrm{x}[l-\Delta l, \mathrm{n}, \mathrm{m}]\circledast \\ h_1[\mathrm{n}, \mathrm{m}]\end{matrix}\right) \quad h_1 \in \mathbb{R}^{\Lambda\times c\times k_1} \tag{4}$$

$$z^{(2)}_{Conv}(c,k_2) = \mathrm{r}(z^{(1)}_{Conv} \circledast h_2[\mathrm{n},\mathrm{m}]) \quad h_2 \in \mathbb{R}^{c\times\Lambda\times k_2}$$

$$z_{flat} = \mathrm{F}\left(z^{(2)}_{Conv}\right) \tag{5}$$

$$z^{(1)}_{Dense}(u) = \mathrm{r}\left(z_{flat}\cdot\omega_1\right) \quad \omega_1 \in \mathbb{R}^{s_F\times u} \tag{6}$$

$$z^{(2)}_{Dense}(u) = \mathrm{r}\left(z^{(1)}_{Dense}(u)\cdot\omega_2\right) \quad \omega_2 \in \mathbb{R}^{u\times s_y} \tag{7}$$

$$\mathbf{m}(X, Y^{gt}, p) = \min\left(\frac{1}{|X|}\sum_{i=0}^{|X|}\left(y_i^{gt} - y_i\right)^2\right) \tag{8}$$

The learning rate was set to 1e-4, and a batch size of 1 was used in the optimizer RMSProp during the training phase. Since the characteristics of solar irradiance trends differ for each season, four models were created to suit the forecasting task for each season.

2) PARAMETER OPTIMIZATION

Selecting kernel sizes, feature channels for convolutional layers, and the number of neurons for dense layers is a common challenge in designing neural networks. Arbitrary selection of these parameters for four different models may introduce design flaws. Instead of arbitrary selection, this paper employed parameter-search optimization using the Bayesian Optimization (BayesOpt) algorithm to automate the neural network design process.

The BayesOpt algorithm [40] is a machine-learning-based optimization method used to search parameters or configurations of a global model aimed at finding a global optimum. It is typically represented as Algorithm 1.

Algorithm 1 Bayesian Optimization for Parameter Selection

Input: $\mathbf{P}$, $\mathbf{m}$
Output: p^*
Prepare p
$\mathcal{P} = \varnothing$
while $s(p;\mathcal{P}) \geqslant 0$ **do**:
  $p' \leftarrow \arg\min_{p\in\mathbf{P}}(a(p;\mathcal{P}))$
  $y \leftarrow \mathbf{m}(p')$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(p', y)\}$
end while
$p^* \leftarrow \min_{(p,y)\in\mathcal{P}} \mathcal{P}(y)$

The BayesOpt Algorithm takes in the classical model function $\mathbf{m}$ which also serves as a fitness function and $p \in \mathbf{P}$ is the latent space for all possible parameter configurations with a dimensionality of $\mathbb{R}^\kappa$. Each parameter configuration p consists of the four parameters $\Delta l, k_1, k_2, c$, and u which then corresponds to κ. $\mathcal{P}$ is then feasible set within $\mathbb{R}^\kappa$ which is initially an empty set before performing the optimization.

$$a(p';\mathcal{P}) = \mathrm{E}\left[\min\mathbf{m}(p')|p', \mathcal{P}\right] - \min\mathbf{m}(p) \tag{9}$$

$$s(p;\mathcal{P}) = \min_{p\in\mathbf{P}}\left[a(p;\mathcal{P}) - \varepsilon\right] \tag{10}$$

Each iteration of the BayesOpt requires selecting a parameter using an acquisition function $a()$. The acquisition function selects a parameter as the difference from the expectation from a conditional probability of the objective and the minimum objective value $\mathbf{m}$ considering the current configuration p. The conditional probability is represented by the minimum objective value of $\mathbf{m}$ when a configuration p' is used, given the prior configurations p' and $\mathcal{P}$. The optimization process is halted when the value of (9) drops below 0. The stopping criterion in (10) identifies the minimum difference between the value of the acquisition function and a threshold ε. The resulting network parameters and the convergence quality of the BayesOpt are reported in Section VI.

D. HYBRID QUANTUM CNN (HQCNN) DESIGN

The classical neural network architectures configured by the BayesOpt algorithm serve as templates for the hybrid quantum neural network architecture, as illustrated in Fig. 8 (b). Convolutional layers are retained as classical layers to maintain the model's performance for high-level feature extraction. The output layer is converted from a classical dense layer to a quantum layer, hence termed a quantum head.

This design is advantageous for assessing the impact of a quantum layer in a hybrid quantum model because the gradient of the quantum layer can be propagated to all classical layers during backpropagation. However, this design is limited by the number of qubits available in current quantum hardware or simulators. The quantum head utilizes only six qubits, as the output vector has a shape of (1, 6). For models where the output vector cannot be represented by the available number of qubits, a dressed quantum layer is suggested. Design of the proposed hybrid quantum CNN (HQCNN) is detailed as follows.

1) ENCODING LAYER

An encoding layer is necessary for the hybrid quantum architecture presented since the input to the quantum layer consists of the classical activations from a preceding classical layer. An R_{YXZ} encoding layer is chosen over Angle and Phase encoding to ensure equal treatment and difficulty for both Pennylane, CUDA Quantum and Torchquantum. This choice is made because R_{YXZ} encoding is not a built-in function in either platform, thereby providing a fair comparison.

$$U_{R_{yxz}}(\phi) = \bigotimes_{i=0}^{N-1} R_y(\phi_i) R_x(\phi_{N+i}) R_z(\phi_{2N+i}) \tag{11}$$

A R_{YXZ} encoding layer can be defined as a parametric unitary $U_{R_{yxz}}$ in (11) that accepts classical data ϕ for N number of qubits. The classical data ϕ should have a length of $3N$ to accommodate a depth of three rotation gates per qubit.

2) ENTANGLEMENT LAYERS

Similarly, to avoid any platform-based biases in forward and backward propagation, the entanglement layer is implemented without using built-in functions. The entanglement

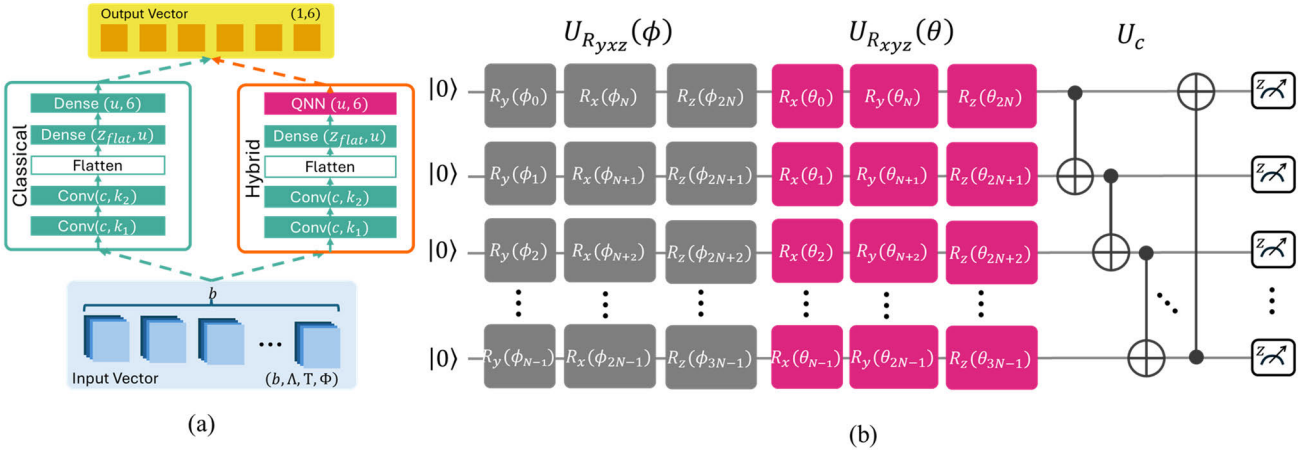

FIGURE 8. (a) General neural network architectures for both classical and hybrid quantum CNNs implementations and (b) the specific design for a quantum neural network layer.

layers are followed by measurement gates in the Z-basis, which observe the quantum information and convert it to classical information.

$$U_{R_{xyz}}(\theta) = \bigotimes_{i=0}^{N-1} R_x(\theta_i) R_y(\theta_{N+i}) R_z(\theta_{2N+i}) \tag{12}$$

$$U_c = \left(\prod_{i=1}^{N-1} \hat{X}_{i-1,i} \right) \hat{X}_{i,0} \tag{13}$$

$$U_q(\theta) = U_{R_{xyz}}(\theta) U_c \tag{14}$$

An entanglement layer is defined using a parametric unitary as shown in (12) and an entanglement unitary as shown in (13). The parametric unitary in (12) is similar to the encoding layer in (11) in terms of arranging parameters for the rotation gates. However, instead of classical data, this unitary accepts the layer weights θ. The entanglement unitary implements a circular entanglement where $\hat{X}_{qi,qj}$ represents a CNOT gate where qi is the control qubit index and qj is the target qubit index.

3) GRADIENT COMPUTATION

A block of a quantum neural network can be represented using (11) and (14). A quantum neural network can then be represented as multiple sequential blocks as seen in (15) with a sequence length R.

$$U(\phi, \theta) = \prod^{R} U_{R_{xyz}}(\phi) U_q(\theta) \tag{15}$$

$$J_\theta \left(\hat{B}_z \right) = U^\dagger(\phi, \theta) \hat{B}_z U(\phi, \theta) \tag{16}$$

The expected value of a quantum neural network in (1) is then revised as (16), considering the entire quantum circuit block (15). It is also noted that a complex64 data type is asserted during gradient computation. Models are coded using the PyTorch Python package for Pennylane, CUDA-Quantum and Torchquantum platforms. Pennylane can readily compute gradients of quantum layers using various methods, such as backpropagation and adjoint methods. The parameter-shift rule is implemented in this paper's hybrid quantum model. Designing quantum-ready hybrid quantum neural networks should consider their execution on quantum resources. As of the writing of this paper, CUDA-Quantum has built-in functionality for the parameter-shift rule in their optimization routines, while Pennylane restricted the use of the parameter-shift rule to non-trainable broadcasted parameters [41]. Hence, a custom parameter-shift rule is implemented to compute the gradient of the quantum layer.

VI. EXPERIMENTAL RESULTS

This section presents the comparative results for training and testing classical CNN and HQCNNs for very short-term solar GHI forecasting. The first part of the result discussion covers the network architecture of the baseline classical neural network, which is subjected to parameter optimization using the Bayesian Optimization Algorithm. The second part of the discussion involves a comparative analysis of the baseline classical models with the hybrid quantum models and a performance comparison of the hybrid quantum platforms for a selected season.

A. NETWORK ARCHITECTURE PARAMETER OPTIMIZATION

As discussed in Section V, parameter optimization is conducted to automate the neural network design process by searching for the values of the kernel sizes, channels, and neuron counts that yield the lowest validation loss, expressed in RMSE.

An initial comparison was conducted with Particle Swarm Optimization (PSO) [46], a commonly used metaheuristic for parameter and hyperparameter optimization. BayesOpt optimized a classical CNN model in 32 minutes over 250 iterations, while PSO required approximately 10 hours. BayesOpt achieved its best RMSE at 2.625%, while PSO reached a slightly lower RMSE of 2.058%. Despite PSO's more optimal RMSE, BayesOpt was chosen for further

model optimization due to its significantly faster convergence, reducing turnaround time in developing foundational architectures for future hybrid quantum models.

The convergence graph for the models of each season is depicted in Fig. 9. A maximum of 200 iterations was set for parameter optimization of each model. Specifically, the spring model converged at 110 iterations with a validation loss of 1.97%, and the summer model converged after 171 iterations with a validation loss of 2.66%, the autumn model converged after 59 iterations with a validation loss of 2.62%, and the winter model converged at 120 iterations with a validation loss of 1.88%.

The autumn model required the fewest optimization iterations, suggesting that the general neural network architecture might be suitable for the autumn data obtained, unlike the summer model, which required the most iterations for optimization. However, it's important to note that initialization conditions may influence optimization speed, as initial configurations closer to the global minima can accelerate convergence. As observed in the solved parameters of each season model in Table 4, the converged losses of the spring and winter models have approximate values, as do those of the summer and autumn models. This pattern is reflected in the determined number of neurons for the dense layers.

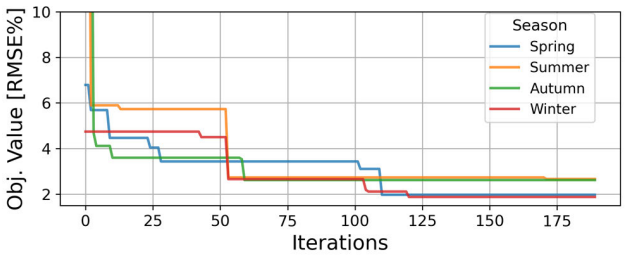

FIGURE 9. Convergence graphs of the parameter optimization of the classical neural networks for all seasons.

TABLE 4. Solved network parameters of models per season.

	Model Parameters				
Season	Δl	k_1	k_2	c	u
Spring	8	2	3	16	94
Summer	7	2	2	10	112
Winter	8	2	3	10	66
Autumn	8	3	3	20	122

B. ARCHITECTURE COMPARISON OF CLASSICAL AND HYBRID QUANTUM MODELS

The performance of the classical and hybrid models for each season is compared based on their training and testing loss. Additionally, the hybrid models are further compared between the platforms they were developed on, namely Pennylane and CUDA-Q.

Fig. 10 indicates that classical models exhibit lower training losses in terms of RMSE compared to hybrid quantum models for autumn, winter, and spring. However, Pennylane shows an average testing loss improvement of 81.54% compared to the classical baseline, while Torchquantum demonstrates a 90.34% improvement, and CUDA-Q achieves an average improvement of 92.30%. This indicates that the classical models may have overfitted the training data for autumn, winter, and spring within just five epochs. Pennylane and Torchquantum exhibit greater training and testing loss among the hybrid quantum platforms than CUDA-Q. This suggests that CUDA-Q may have a more optimized implementation of quantum algorithms than Pennylane and Torchquantum. A more conclusive test will be presented in the succeeding sub-section.

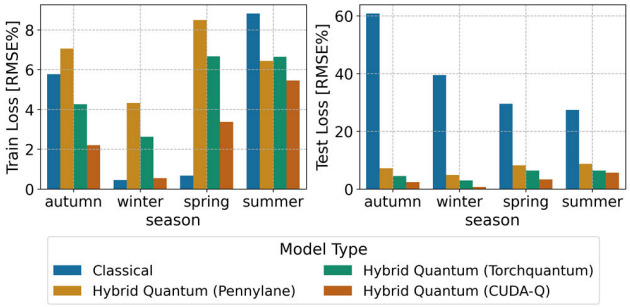

FIGURE 10. Per season RMSE (%) comparison of classical and hybrid quantum neural networks.

Further tests using the best hybrid quantum model were conducted by evaluating test data loss on a monthly basis, as shown in Fig. 11. It can be observed that the loss for classical models is generally higher than that of the hybrid quantum counterpart. Both models exhibit increased losses during the seasonal transition months—February, May, September, and November—corresponding to the transitions between winter-spring, spring-summer, summer-autumn, and autumn-winter. These months in Taiwan are characterized by significant temperature fluctuations and cloud movement, which impact the accuracy of predictions.

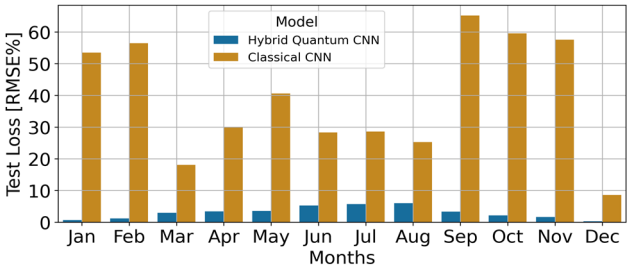

FIGURE 11. Per Month RMSE (%) comparison between classical CNN and proposed HQCNN.

Further experiments were conducted on other deep neural network models to evaluate the advantages of hybrid quantum models. Foundational classical architectures, such as ANN and LSTM, were selected as additional benchmarks for simplicity. Additional tests were performed using hybrid quantum ANNs with the dimensionally reduced dataset. As shown in Fig. 12, LSTM outperforms the other classical

models. Comparing ANN and PCA+ANN, the dimensionally reduced model using PCA shows only a slight increase in RMSE%. However, consistent with Fig. 10, the hybrid HQNN and the proposed hybrid quantum CNNs (HQCNN) generally outperform the foundational classical models, indicating that the inclusion of quantum layers significantly reduces RMSE values.

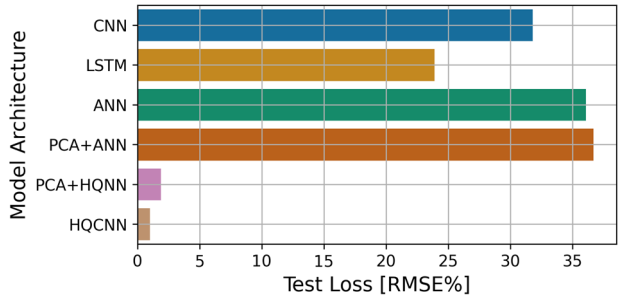

FIGURE 12. RMSE comparison of classical and hybrid quantum neural networks.

C. ROBUSTNESS ASSESSMENT

The robustness of the proposed HQCNN and classical CNN models was evaluated using simulated sensor fault scenarios from the modified test set. The experiment incorporated sensor/environment noise, no measurement, and spiking measurement fault scenarios, as shown in Table 5. The test set was modified by introducing noise to simulate common sensor measurement issues. Sensor or environment noise was modeled by adding Gaussian noise to the GHI feature, using a Gaussian distribution with varying noise standard deviations (σ). Zero or no measurement scenarios were simulated using a modified dropout routine, where a frequency parameter (f_z) indicates how often data fails to be recorded. This frequency ranges from 0.0 (no missing data) to 1.0 (complete data loss). A Bernoulli function applied to f_z creates a binary mask that simulates zero measurements in the GHI data. Spiking or glitching scenarios, which may result from external disturbances or electrostatic discharges, were simulated using a random spiking function applied to the GHI data at a given spiking frequency (f_s).

The summer season test set was used for the experiments. Three values of σ (0.05, 0.1, and 0.15) were considered for sensor/environment noise scenarios to simulate varying degrees of information loss. For zero measurement scenarios, frequencies f_z of 0.05, 0.1, and 0.2 were used, while spiking measurement scenarios employed frequencies f_s of 0.02, 0.04, and 0.08. The sensitivity (s) of the model to noise is detailed in Table 5, where it is calculated as the ratio of the error with the noisy modified test set to the error with the original test set.

Both classical CNN and proposed HQCNN are expected to experience degradation in prediction quality. Observations show that classical CNNs generally have higher sensitivity, averaging 11.82, compared to HQCNNs, which average 2.5. This suggests that the quantum layers in HQCNNs may contribute to improved feature learning and generalization. The spiking measurement scenarios impact both classical CNN and HQCNN models the most, highlighting the severity of these scenarios in obscuring predictions. Although HQCNNs generally exhibit lower sensitivity, their sensitivity increases linearly with the intensity of different noise types, unlike classical models, which show only variable sensitivity.

D. PLATFORM PERFORMANCE COMPARISON

Additional experiments detailing the advantages of each hybrid quantum platform are presented in this subsection. The effects of using hardware acceleration are also included in the succeeding discussions. Further, the training and testing of the HQCNNs are done with a hybrid CPU and GPU setup. The experiments primarily focus on testing GPU workstations or high-performance computers, specifically for GPU-based quantum circuit simulations. The hardware setup used in this experiment comprises a GPU workstation equipped with an 11th Gen. Intel i7-11700 processor, NVIDIA RTX 3070 GPUs, and 32 GB of RAM. The summer dataset is representative to facilitate a standard comparison for different model setups. All models are trained for only 5 epochs to assess their efficiency for short training periods. To ensure fairness, batch sizes were set to 1 for Pennylane, CUDA-Q and Torchquantum's training pipelines, resulting in each epoch looping over 145,728 samples.

Pennylane is widely utilized in numerous applied research involving hybrid quantum neural networks. Table 6 provides a performance analysis using Pennylane for the HQCNN utilized in this paper. Both neural network optimization and quantum circuit state vector simulation are conducted with various combinations of CPU and GPU modalities. Several GPU acceleration libraries are compared alongside CPU-based computation. While CPU-based computations do not utilize any acceleration libraries, GPU-based computations may invoke CUDA, CUDA with cuDNN for GPU acceleration in classical neural network layer operations, and cuQuantum for state vector computations.

Both loss and timing performance changes are observed between CPU and GPU-based operations. An approximate average testing loss improvement of 10.20% is seen between models that utilize cuQuantum as an accelerator for quantum layers compared to those that do not. Additionally, improvements in timing performances, such as total training time and inference time, are noted. Using accelerators for classical layers leads to an expected decrease in total training and inference time. However, combining classical and quantum layer acceleration results in significant latency reduction, with a maximum speedup of approximately 275% for average latency per epoch and approximately 218% for inference time. This suggests that most of the complexity of the HQCNN training is attributed to operations related to the quantum neural network, such as forward and backward propagation. Similar to most deep learning applied research, CUDA, and cuDNN demonstrate improvements in timing and

TABLE 5. Robustness comparison between classical and hybrid quantum CNNs.

Model	Test Cases								
	Sensor/Environment Noise (Gaussian) RMSE% / s			Measurement Error (No reading) RMSE% / s			Measurement Error (Spiking) RMSE% / s		
	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.15$	$f_z = 0.05$	$f_z = 0.1$	$f_z = 0.2$	$f_s = 0.02$	$f_s = 0.04$	$f_s = 0.08$
Classical CNN	66.030% / 11.69	66.601% / 11.79	66.564% / 11.78	66.709% / 11.81	66.941% / 11.85	67.077% / 11.87	66.589% / 11.79	67.544% / 11.95	67.274% / 11.91
HQCNN	**5.003% / 2.54**	**5.001% / 2.54**	**5.008% / 2.55**	**4.401% / 2.24**	**4.642% / 2.36**	**4.811% / 2.45**	**5.046% / 2.57**	**5.084% / 2.59**	**5.149% / 2.62**

TABLE 6. Pennylane performance analysis.

Device (Classical/Quantum)	Classical CNN GPU Acceleration	Quantum NN GPU Acceleration	Test RMSE (%)	Ave. Latency per Epoch (s)	Total Training Time (min)	Inference Time (ms)
CPU/CPU	None	None	1.93%	23,183	1,932	26.33
GPU/CPU	CUDA	None	1.96%	20,553	1,713	25.19
GPU/CPU	cuDNN	None	1.89%	19,840	1,653	24.88
CPU/GPU	None	cuQuantum	2.00%	6,297	525	8.78
GPU/GPU	CUDA	cuQuantum	1.64%	6,189	516	8.00
GPU/GPU	cuDNN	cuQuantum	**1.55%**	**5,281**	**441**	**7.80**

TABLE 7. Performance comparison among Pennylane, Torchquantum and CUDA-Q.

Platform	Classical CNN GPU Acceleration	Test RMSE (%)	Ave. Latency per Epoch (s)	Total Training Time (min)	Inference Time (ms)
Pennylane	None	2.00%	6,297	525	8.78
	CUDA	1.64%	6,189	516	8.00
	cuDNN	1.55%	5,281	441	7.80
Torchquantum	None	6.67%	34,937	2,911	47.72
	CUDA	5.02%	62,224	5,185	81.01
	cuDNN	4.26%	61,714	5,142	82.93
CUDA-Q	None	2.70%	2,457	205	3.89
	CUDA	1.00%	1,921	160	2.65
	cuDNN	**0.45%**	**1,909**	**159**	**2.65**

loss performances compared to implementations that utilize only CPU for neural network operations.

The performance of CUDA-Q is compared with those of Pennylane and Torchquantum for the same dataset, as shown in Table 7. The column for GPU acceleration of the quantum layers is omitted, as all experiments involve GPU-based acceleration in state vector simulation using cuQuantum. A noticeable difference in timing performance is observed between Pennylane and CUDA-Q. CUDA-Q is 2.7 times faster than Pennylane for training and 2.9 times faster for inference. CUDA-Q is also 32.34 times faster for training and 31.29 times faster for inference compared to Torchquantum. The classical layers of the models implemented in Pennylane and CUDA-Q are designed and developed similarly in PyTorch. Therefore, it can be inferred that CUDA-Q can compute forward and backward propagation for the quantum layer more efficiently than Pennylane, considering complex64-based data computation. A similar improvement can be observed in loss performance when using acceleration on both classical and quantum layers. An observed 39.02% and 70.97% testing loss improvement is seen for pipelines running CUDA and CUDA+cuDNN, respectively.

Pennylane offers a convenient platform for designing and implementing variational quantum circuits and quantum neural networks, allowing for more flexible ways to prototype variational quantum circuit formulations. Additionally, Torchquantum needs to improve its implementation of the parameter-shift rule for gradient computation and enhance its use of GPU acceleration for statevector simulations. On the other hand, CUDA-Q has not yet implemented abstracted functions and APIs for high-level quantum circuit design. However, experimental results demonstrate that CUDA-Q outperforms both Pennylane and Torchquantum in terms of loss and timing performances for training and testing HQCNNs. This improvement may be attributed to the optimized compiler toolchain for the quantum kernel execution and additional GPU memory allocation techniques and data structures implemented in CUDA-Q.

VII. CONCLUSION

This work compares the design and implementation of hybrid quantum neural networks (HQNNs) across development platforms for GPU-based operations pipelines. This comparison is demonstrated using a multi-location, very short-term solar GHI forecasting model as a use case. The main

contributions of this paper can be summarized as follows: (a) A statistics-driven classical neural network architecture with further parameter optimization by Bayesian Optimization is presented. (b) A performance comparison of hybrid quantum neural network architectures and classical neural networks is presented. (c) A comparative analysis of hybrid quantum deep learning platforms considering their loss and latency performances is presented.

In summary, the experimental results demonstrate the following: (a) The initial statistics-driven approach for neural network model design, combined with Bayesian Optimization for design refinement, offers an automated method for developing localized classical neural networks for the northwestern Taiwan region, resulting in low validation losses. The summer model required the highest number of iterations to converge at 171, while the autumn model needed the least, at 59 iterations. (b) When comparing classical and hybrid quantum neural networks, hybrid quantum neural networks exhibit lower testing losses. Specifically, Pennylane achieves an average of 81.54% testing loss improvement, while Torchquantum achieves 90.34%, and CUDA-Q achieves an average of 92.30% testing loss improvement. (c) Pennylane's performance, when considering hardware acceleration, shows that GPU-based computations with cuDNN and cuQuantum provide significant speedup in training and testing times. This indicates that GPU acceleration libraries enhance efficiency for hybrid quantum operations. (d) Comparing the performance of Pennylane, Torchquantum, and CUDA-Q, CUDA-Q shows significant improvements over Pennylane in terms of testing loss, training time, and inference time. This suggests that CUDA-Q has the potential for deployed operational purposes.

The comparative study demonstrates that using GPU and GPU hardware acceleration platforms can significantly improve performance over classical implementations for multi-location, very short-term solar GHI forecasting. Future enhancements to this study could include exploring different neural network architectures, such as attention-based models. Additionally, deployment strategies for quantum-enhanced and quantum computer-integrated solutions could be investigated. These considerations highlight potential areas for future research and development in the operability of systems involving hybrid quantum algorithms.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA Corporation, especially NVIDIA AI Technology Center (NVAITC) in Taiwan, for providing technical support and comments on this work through the NVIDIA-CYCU AI University platform.

REFERENCES

[1] F. Egli, "Renewable energy investment risk: An investigation of changes over time and the underlying drivers," *Energy Policy*, vol. 140, May 2020, Art. no. 111428.

[2] S. M. J. Jalali, S. Ahmadian, A. Kavousi-Fard, A. Khosravi, and S. Nahavandi, "Automated deep CNN-LSTM architecture design for solar irradiance forecasting," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 52, no. 1, pp. 54–65, Jan. 2022.

[3] X. Jiao, X. Li, D. Lin, and W. Xiao, "A graph neural network based deep learning predictor for spatio-temporal group solar irradiance forecasting," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6142–6149, Sep. 2022.

[4] S. Tajjour, S. S. Chandel, M. A. Alotaibi, H. Malik, F. P. García Márquez, and A. Afthanorhan, "Short-term solar irradiance forecasting using deep learning techniques: A comprehensive case study," *IEEE Access*, vol. 11, pp. 119851–119861, 2023.

[5] Z. Zhen, J. Liu, Z. Zhang, F. Wang, H. Chai, Y. Yu, X. Lu, T. Wang, and Y. Lin, "Deep learning based surface irradiance mapping model for solar PV power forecasting using sky image," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 3385–3396, Jul. 2020.

[6] Y. Yu, G. Hu, C. Liu, J. Xiong, and Z. Wu, "Prediction of solar irradiance one hour ahead based on quantum long short-term memory network," *IEEE Trans. Quantum Eng.*, vol. 4, pp. 1–15, 2023.

[7] Y. Y. Hong and J. B. D. Santos, "Day-ahead spatiotemporal wind speed forecasting based on a hybrid model of quantum and residual long short-term memory optimized by particle swarm algorithm," *IEEE Syst. J.*, vol. 17, no. 4, pp. 1–12, Dec. 2023.

[8] Y.-Y. Hong, C. J. E. Arce, and T.-W. Huang, "A robust hybrid classical and quantum model for short-term wind speed forecasting," *IEEE Access*, vol. 11, pp. 90811–90824, 2023.

[9] A. Sagingalieva, "Photovoltaic power forecasting using quantum machine learning," 2023, *arXiv:2312.16379*.

[10] A. Safari and A. A. Ghavifekr, "Quantum technology & quantum neural networks in smart grids control: Premier perspectives," in *Proc. 8th Int. Conf. Control, Instrum. Autom. (ICCIA)*, Mar. 2022, pp. 1–6.

[11] A. Ajagekar and F. You, "Quantum computing for energy systems optimization: Challenges and opportunities," *Energy*, vol. 179, pp. 76–89, Jul. 2019.

[12] D. P. DiVincenzo, D. Bacon, J. Kempe, G. Burkard, and K. B. Whaley, "Universal quantum computation with the exchange interaction," *Nature*, vol. 408, no. 6810, pp. 339–342, Nov. 2000.

[13] D. S. Prasad and I. Ahmad, "Quantum computing basics, applications and future perspectives," *J. Mol. Struct.*, vol. 1308, Jul. 2024, Art. no. 137917.

[14] M. S. Salek, P. K. Biswas, J. Pollard, J. Hales, Z. Shen, V. Dixit, M. Chowdhury, S. M. Khan, and Y. Wang, "A novel hybrid quantum-classical framework for an in-vehicle controller area network intrusion detection," *IEEE Access*, vol. 11, pp. 96081–96092, 2023.

[15] S.-G. Jeong, Q.-V. Do, H.-J. Hwang, M. Hasegawa, H. Sekiya, and W.-J. Hwang, "Hybrid quantum convolutional neural networks for UWB signal classification," *IEEE Access*, vol. 11, pp. 113726–113739, 2023.

[16] A. Chalumuri, R. Kune, S. Kannan, and B. S. Manoj, "Quantum–classical image processing for scene classification," *IEEE Sensors Lett.*, vol. 6, no. 6, pp. 1–4, Jun. 2022.

[17] J. Zhang, J. Tian, M. Li, J. I. Leon, L. G. Franquelo, H. Luo, and S. Yin, "A parallel hybrid neural network with integration of spatial and temporal features for remaining useful life prediction in prognostics," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[18] A. P. Ngo, N. Le, H. T. Nguyen, A. Eroglu, and D. T. Nguyen, "A quantum neural network regression for modeling lithium-ion battery capacity degradation," in *Proc. IEEE Green Technol. Conf.*, Apr. 2023, pp. 164–168.

[19] D. Balakrishnan, U. Mariappan, P. G. M. Raghavendra, P. K. Reddy, R. L. N. Dinesh, and S. B. Jabiulla, "Quantum neural network for time series forecasting: Harnessing quantum Computing's potential in predictive modeling," in *Proc. 2nd Int. Conf. Futuristic Technol. (INCOFT)*, vol. 19, Nov. 2023, pp. 1–7.

[20] H. E. Brandt, "Qubit devices and the issue of quantum decoherence," *Prog. Quantum Electron.*, vol. 22, nos. 5–6, pp. 257–370, Sep. 1999.

[21] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature Commun.*, vol. 11, no. 1, p. 808, Feb. 2020.

[22] K. Poland, K. Beer, and T. J. Osborne, "No free lunch for quantum machine learning," 2020, *arXiv:2003.14103*.

[23] K. Beer, M. Khosla, J. Köhler, and T. J. Osborne, "Quantum machine learning of graph-structured data," 2021, *arXiv:2103.10837*.

[24] M. M. Sushmit, "Forecasting solar irradiance with hybrid classical-quantum models: A comprehensive evaluation of deep learning and quantum-enhanced techniques," *Energy Convers. Manage.*, vol. 294, p. 15, Jun. 2023, Art. no. 117555.

[25] A. Shadab, S. Ahmad, and S. Said, "Spatial forecasting of solar radiation using ARIMA model," *Remote Sens. Appl. Soc. Environ.*, vol. 20, Nov. 2020, Art. no. 100427.

[26] R. GhoshThakur, A. Basu, Z. Haque, B. Bhattacharya, S. GonChaudhuri, and S. Balachandran, "Performance prediction of the micro solar dome in different climatic regions of India from pilot-scale by random forest algorithm," *Sustain. Energy Technol. Assessments*, vol. 52, Aug. 2022, Art. no. 102163.
[27] M. J. Sammar, M. A. Saeed, S. M. Mohsin, S. M. A. Akber, R. Bukhsh, M. Abazeed, and M. Ali, "Illuminating the future: A comprehensive review of AI-based solar irradiance prediction models," *IEEE Access*, vol. 12, pp. 114394–114415, 2024.
[28] X. Hou, C. Ju, and B. Wang, "Prediction of solar irradiance using convolutional neural network and attention mechanism-based long short-term memory network based on similar day analysis and an attention mechanism," *Heliyon*, vol. 9, no. 11, Nov. 2023, Art. no. e21484.
[29] S. Sharda, M. Singh, and K. Sharma, "RSAM: Robust self-attention based multi-horizon model for solar irradiance forecasting," *IEEE Trans. Sustain. Energy*, vol. 12, no. 2, pp. 1394–1405, Apr. 2021.
[30] H. Zhang, B. Li, S.-F. Su, W. Yang, and L. Xie, "A novel hybrid transformer-based framework for solar irradiance forecasting under incomplete data scenarios," *IEEE Trans. Ind. Informat.*, vol. 20, no. 6, pp. 8605–8615, Jun. 2024.
[31] D. Emmanoulopoulos and S. Dimoska, "Quantum machine learning in finance: Time series forecasting," 2022, *arXiv:2202.00599*.
[32] V. Bergholm, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2018, *arXiv:1811.04968*.
[33] M. Malvoni, M. G. De Giorgi, and P. M. Congedo, "Data on photovoltaic power forecasting models for Mediterranean climate," *Data Brief*, vol. 7, pp. 1639–1642, Jun. 2016.
[34] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Sci. Technol.*, vol. 4, no. 4, Nov. 2019, Art. no. 043001.
[35] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Phys. Rev. A, Gen. Phys.*, vol. 98, no. 3, Sep. 2018, Art. no. 032309.
[36] S. Chetlur, "CuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*.
[37] H. Bayraktar et al., "CuQuantum SDK: A high-performance library for accelerating quantum science," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2023, pp. 1050–1061.
[38] I. A. Luchnikov, O. E. Tatarkin, and A. K. Fedorov, "High-performance state-vector emulator of a quantum computer implemented in the rust programming language," in *AIP Conf. Proc.*, vol. 2948, 2023, p. 20.
[39] *IBM Quantum Documentation*. Accessed: Jun. 1, 2024. [Online]. Available: https://docs.quantum.ibm.com/guides
[40] Google Quantum AI. *Google Cirq: A Python Open Source Library for Quantum Computing*. Accessed: May 1, 2024. [Online]. Available: https://quantumai.google/cirq
[41] K. M. Svore et al., "Q#: Enabling scalable quantum computing and development with a high-level DSL," in *Proc. Real World Domain Specific Lang. Workshop*, 2018, pp. 1–10, Art. no. 7. Accessed: May 1, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3183895.3183901
[42] M. Broughton et al., "TensorFlow quantum: A software framework for quantum machine learning," 2020, *arXiv:2003.02989*.
[43] J.-S. Kim, A. McCaskey, B. Heim, M. Modani, S. Stanwyck, and T. Costa, "CUDA quantum: The platform for integrated quantum-classical computing," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2023, pp. 1–4.
[44] H. Wang, Z. Liang, J. Gu, Z. Li, Y. Ding, W. Jiang, Y. Shi, D. Z. Pan, F. T. Chong, and S. Han, "TorchQuantum case study for robust quantum circuits," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Oct. 2022, pp. 1–9.
[45] M. Sengupta, Y. Xie, A. Lopez, A. Habte, G. Maclaurin, and J. Shelby, "The national solar radiation data base (NSRDB)," *Renew. Sustain. Energy Rev.*, vol. 89, pp. 51–60, Jun. 2018.
[46] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95-Int. Conf. Neural Networks. Nov.*, 1995, pp. 1942–1948.

YING-YI HONG (Senior Member, IEEE) received the B.S.E.E. degree from Chung Yuan Christian University (CYCU), Taiwan, in 1984, the M.S.E.E. degree from National Cheng Kung University (NCKU), Taiwan, in 1986, and the Ph.D. degree from the Department of Electrical Engineering, National Tsing Hua University (NTHU), Taiwan, in December 1990. Sponsored by the Ministry of Education, Taiwan. He conducted research with the Department of Electrical Engineering, University of Washington, Seattle, WA, USA, from August 1989 to August 1990. Since 1991, he has been with CYCU, where he was the Dean of the College of Electrical Engineering and Computer Science, from 2006 to 2012. He was promoted to the Chair Professor due to his exceptional research, leadership, teamwork, and international collaboration, in 2024. From 2012 to 2018, he was a General Secretary with CYCU, where he is currently the Vice President. His research interests include power system analysis and artificial intelligence applications. He received the Outstanding Professor of the Electrical Engineering Award from the Chinese Institute of Electrical Engineering (CIEE), Taiwan, in 2006, and the Outstanding Professor of the Engineering Award from the Chinese Institute of Engineers (CIE), Taiwan, in 2024. He was the Chair of the IEEE PES Taipei Chapter, in 2001.

DYLAN JOSH DOMINGO LOPEZ (Member, IEEE) received the B.S. degree in computer engineering degree from Adamson University, Philippines, in 2018, and the M.Sc. degree in computer science and information engineering from Chung Hua University, Taiwan, in 2020. He is currently the Ph.D. degree in EE with CYCU. He is a part-time Associate Professor with the Technological Institute of the Philippines, Quezon, and an Instructor with De La Salle University, Manila, Philippines. His research interests include optimization, quantum computing, and artificial intelligence applications.

YUN-YUAN WANG received the bachelor's degree in physics from National Tsing Hua University (NTHU) and the master's degree from the College of Electrical Engineering and Computer Science, National Taiwan University (NTU), in 2019. Currently, he is a Senior Solution Architect with the NVIDIA AI Technology Center (NVAITC), specializing in quantum computing and GPU acceleration. He also holds a total of 11 licensed U.S. patents, covering quantum-inspired algorithms, parallel computing architecture, and semiconductor technologies.