IEEE Transactions on Quantum Engineering

Received 17 September 2024; revised 8 August 2025; accepted 15 December 2025; date of publication 18 December 2025; date of current version 28 January 2026.

Digital Object Identifier 10.1109/TQE.2025.3646040

Integrated Encoding and Quantization to Enhance Quanvolutional Neural Networks

DANIELE LIZZIO BOSCO[1,2], BEATRICE PORTELLI[1,3], AND GIUSEPPE SERRA[1]

[1]Department of Mathematics, Computer Science and Physics, University of Udine, 33100 Udine, Italy
[2]Department of Biology, University of Naples Federico II, 80126 Naples, Italy
[3]Department of Agriculture, Animal, Environmental and Food Sciences, University of Udine, 33100 Udine, Italy

Corresponding author: Daniele Lizzio Bosco (e-mail: lizziobosco.daniele@spes.uniud.it).

ABSTRACT Image processing is one of the most promising applications for quantum machine learning. Quanvolutional neural networks with nontrainable parameters are the preferred solution to run on current and near future quantum devices. The typical input preprocessing pipeline for quanvolutional layers comprises of four steps: optional input binary quantization, encoding classical data into quantum states, processing the data to obtain the final quantum states, and decoding quantum states back to classical outputs. In this article, we propose two ways to enhance the efficiency of quanvolutional models. First, we propose a flexible data quantization approach with memoization, applicable to any encoding method. This allows us to increase the number of quantization levels to retain more information or lower them to reduce the amount of circuit executions. Second, we introduce a new integrated encoding strategy, which combines the encoding and processing steps in a single circuit. This method allows great flexibility on several architectural parameters (e.g., number of qubits, filter size, and circuit depth) making them adjustable to quantum hardware requirements. We compare our proposed integrated model with a classical convolutional neural network and the well-known rotational encoding method, on two different classification tasks. The results demonstrate that our proposed model encoding exhibits a comparable or superior performance to the other models while requiring fewer quantum resources.

INDEX TERMS Convolutional neural networks (CNNs), image processing, noisy intermediate-scale quantum (NISQ), quantum computing, quantum encoding, quantum machine learning (QML), quanvolutional neural network (QuanvNN).

I. INTRODUCTION

The field of quantum machine learning (QML) applied to computer vision has gathered increasing interest in the last decade, combining quantum computing and machine learning to develop new algorithms, which may lead to more efficient and optimized computer vision models [1]. A promising approach in image processing is the application of quantum convolutional neural networks (CNNs), also known as quanvolutional neural networks (hereafter QuanvNN s), which aim to enhance classical models through hybrid (classical–quantum) architectures. However, current quantum devices are still characterized by a limited number of qubits and the absence of error correction. This hinders QML from matching the performance of classical machine learning methods. Therefore, the applications of QuanvNNs are currently limited to simple architectures and small datasets because of several constraints, including the low number of available qubits, the need to reduce circuit depth to avoid decoherence, and technical optimization constraints. For instance, while deep learning relies on gradient descent for parameter updates, quantum neural networks (QNNs) require the use of the parameter shift rule [2], which involves a large amount of additional circuit measurements and is unreliable in the absence of error correction. This challenge, commonly referred to as the barren plateau phenomenon [3], affects most variational algorithms that rely on quantum hardware to compute the loss function. This makes large-scale QuanvNN optimization currently impractical.

To avoid the need to optimize quantum circuit parameters, a solution is the use of quanvolutional layers with

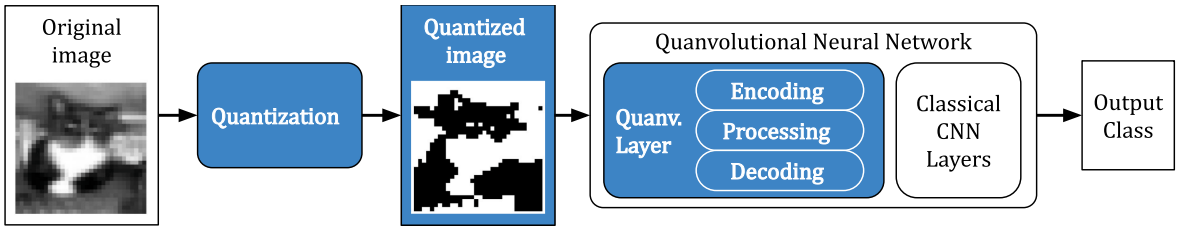

FIGURE 1. Schema of the full pipeline using a quanvolutional neural network. The input image is first quantized and then passed to the network, which comprises a quanvolutional layer (encoding, processing, and decoding) and one or more classical convolutional layers that provide the final output.

nontrainable parameters, as introduced in [4]. These layers can be used in hybrid models for preprocessing the input dataset, acting as random feature extractors that might identify features that are challenging to compute using classical methods. Some examples of successful applications, where quantum methods employing trainable quantum parameters would be impractical on current quantum or simulated devices, include speech recognition [5], building damage assessment [6], medical diagnostics [7], and pollution emission monitoring [8].

The key components of QuanvNNs are the quanvolutional layers, the quantum equivalent of the classical convolutional layers, which process the input in a locally invariant fashion, detecting patterns and extracting meaningful information. Typically, quanvolutional layers with nontrainable parameters are the first layers of a QuanvNN, and their input preprocessing comprises the following stages (see Fig. 1).

1) *Binary quantization:* If needed, the original input image is converted to a new image with binary pixel values.
2) *Encoding:* The classical input is transformed into quantum states.
3) *Processing:* The quantum states are processed through a predefined quantum circuit to create interaction between the input features.
4) *Decoding:* Classical information is extracted from the final quantum state of the processing circuit.

Quanvolutional layers still present some challenges that need to be addressed.

For example, binary quantization simplifies the input data to decrease the number of required quantum circuit executions; however, it may lead to considerable information loss for some tasks, which rely on fine-grained information. While some encodings do not require quantization and work on the original image, several strategies need binary quantization as a prerequisite, such as threshold encoding [4].

On the other hand, encoding and processing stages are the main source of the quantum hardware requirements for QuanvNNs. For example, rotational encoding [9] requires a number of qubits, which is quadratic with respect to the size of the input patch, usually denoted as kernel size. As an example, a kernel of size 5 requires 25 qubits. Other encodings are even more resource expensive, such as amplitude encoding [10], [11] which requires an exponential number of gates to prepare the quantum states. These techniques are unsuitable for noisy intermediate-scale quantum (NISQ) devices, both for the required number of input qubits and their circuit depth, which may cause problems due to limited coherence time.

To address these challenges, we propose a new quanvolutional layer that makes the quantization, encoding, and processing stages more resource efficient and NISQ-friendly while retaining the performance of more expensive quanvolutional models.

As regards quantization step, we implement a flexible quantization approach coupled with a lookup table. The number of quantization levels can be analytically chosen based on the amount of information loss on the input data. This allows increasing (or decreasing) the number of quantization levels as needed depending on the application, deciding between retaining more information (more quantization levels) or lowering the number of circuit executions and measurements (fewer quantization levels).

With respect to the encoding and processing stages, we propose a quanvolutional model that, differently from previous ones, integrates both steps in a single circuit. The proposed integrated encoding has no constraints on the number of qubits and gates needed to process the input and can, therefore, be adapted to the available resources. This encoding allows creating small and efficient models that are NISQ-friendly, but also scaling up the number of qubits and gates as new resources become available.

We compare the proposed integrated model with a quanvolutional model using rotational encoding introduced in [4] and used in [5], [6], [7], [8], [12], and [13], as well as a classical CNN model. The performance of all models is tested on two different datasets for binary (MiraBest [14]) and multiclass image classification (Liquid Argon Time Projection Chamber (LArTPC) [15]), using different circuit configurations and testing different kernel sizes for the input patches.

Our experiments show that the proposed integrated encoding surpasses the performance of the traditional rotational encoding for all kernel sizes. In addition, integrated encoding is more resource-efficient and could, therefore, be tested on larger kernel sizes compared to rotational encoding. Finally, our proposed approach outperforms the classical CNN on

both tasks, while the model using rotational encoding only does so on the binary classification benchmark.

In addition, our experiments show that, contrary to standard quanvolutional models, the expressivity of our proposed integrated model changes with the number of gates used. In the literature, expressivity has been associated with improved performance in QNNs for classification tasks [16], but also with challenges in trainability for parameterized QNNs, such as the occurrence of barren plateaus [17]. In our case, however, we did not observe a strong or consistent correlation between model expressivity and classification accuracy.

To encourage further research on this topic, we share all the code to reproduce our experiments,[1] including the implementation of the quantization procedure and the quanvolutional model with integrated encoding.

The rest of this article is organized as follows. In Section II, we present a review of the existing literature on the topic. In Section III, we provide a concise background of the technical concepts used in this work, including quanvolutional layers and standard encoding strategies. In Section IV, we introduce our proposed pipeline, describing the data quantization method and the proposed integrated encoding. Section V contains the details of the experimental setting, including datasets, tested models, and implementation details. Section VI presents and discusses the experimental results. Finally, Section VII concludes this article.

II. RELATED WORK

Previous research has developed two main classes on quanvolutional methods: the ones where quantum circuit parameters have learnable parameters, and the methods where they have fixed nonlearnable parameters (i.e., only the classical parameters are learned).

A. QUANVNNS WITH LEARNABLE PARAMETERS

These quantum convolutional models include parameterized circuits, which can be trained together with the classical parameters via the parameter shift rule or similar techniques. This approach allows the quantum circuit to learn representations tailored to the task, potentially capturing richer nonclassical features.

The most common approach consists in encoding the input image in a quantum state, processing it via one or more quantum convolutional layers, decoding the final state, and finally passing it to a classical architecture.

For example, Chen et al. [15] implemented this kind of QuanvNN for the classification of high energy physics particles on the LArTPC dataset. Their model encodes each pixel of a 2×2 patch in a qubit by converting it in rotation angles. The processing circuit has four qubits and a fixed architecture that contains parameterized rotations. These parameters are initialized randomly and iteratively optimized during training. The output of the quantum circuit is then decoded and passed to a series of classical fully connected layers, which are jointly trained with the quantum circuit.

Other works apply the same architecture to the classification of 2-D radiological images [12] and several multiclass classification tasks on MNIST, Medical MNIST, and CIFAR-10 [13]. The former tests the effectiveness of several encoding methods that require one qubit per pixel (threshold encoding, rotational encoding, and higher order encoding), while the latter uses rotational encoding.

Another interesting approach consists in placing the quantum circuit between two classical layers. For example, Matic et al. [12] also introduced a model for the classification of 3-D radiological images, which takes $2 \times 2 \times 2$ patches as input. The model consists of two classical convolutional layers, followed a quantum convolutional layer with eight filters. Each filter uses eight qubits, angle encoding, and contains gates with trainable rotations. Finally, the output of the circuit is processed by further classical fully connected layers as seen above.

Similar architectures are employed in [13] and [18], where the quantum circuit is preceded by a classical "embedding" phase. This approach has the advantage of decoupling the dimension of the input patches from the dimensions of the circuit: since the input image is first processed by classical layers, these can be used to create a low-dimensional embedding, which requires less quantum resources to encode and process. On the other hand, this process does not allow the quantum circuit to access the real input data. This may hamper the ability of the quantum circuit to detect meaningful nonclassical patterns.

Notably, models with trainable quantum parameters typically require quantum hardware in the loop during training and are thus considered more resource-intensive.

B. QUANVNNS WITH NONLEARNABLE PARAMETERS

To reduce quantum hardware requirements, several works explore QuanvNNs with nontrainable quantum circuits. In these models, the quantum parameters are usually randomly initialized and remain fixed, while the classical layers of the network undergo the standard training procedure. Another advantage of using a nonlearnable quanvolutional layer is that its output can be precomputed for all the input patches in the dataset, further reducing the amount of quantum circuit executions required to train the model.

This kind of QuanvNNs was first introduced by Henderson et al. [4]. In this work, the authors use a classical CNN preceded by a quanvolutional layer. The authors use threshold encoding on 3×3 input patches, meaning that the processing circuit requires nine qubits. The processing circuit is composed of a sequence of randomly selected parametric and nonparametric gates (more details in Section III-C). Any required gate parameter is also selected randomly. The authors test networks consisting of 1–50 quanvolutional filters, showing that the performance of the QuanvNN increases with the number of filters, but this effect caps at around 25 filters.

[1] https://github.com/Dan-LB/integrated_encoding_for_QuanvNN

Following works used similar architectures for the assessment of seismic damage from photos [6] and the detection of arrhythmia from ECG signals [7]. Both works employed rotational encoding instead of threshold encoding, allowing to encode a wider range of information. They also used smaller 2×2 input patches, resulting in a quantum circuit with four qubits, followed by classical convolutional and fully connected layers.

Finally, Mattern et al. [19] compare QuanvNNs models with randomly generated circuits with both learnable and nonlearnable parameters. The authors compare the models created from all possible combinations of three different encodings (flexible representation of quantum images (FRQI), novel enhanced quantum image representation (NEQR), and threshold encoding), two input patch sizes (2×2 with two filters and 4×4 with four filters, leading to a qubit requirement of 3 and 16, respectively), and two settings for the parameters (learnable and nonlearnable). The final results show that both learnable and nonlearnable circuits can achieve high performances on the MNIST dataset (over 0.82 accuracy). In addition, larger input patch sizes do not necessarily lead to higher performances, as all tested models reached higher accuracy when using 2×2 patches.

The QuanvNN models used in this work are also based on randomly generated nonlearnable quantum circuits. Differently to previous works, we further test the effect of using different input patch sizes (from 2×2 to 5×5). In addition, we compare the performance of the traditional combination of rotational encoding and Henderson-based processing circuit with the proposed integrated encoding, which shows increased performance with less quantum requirements.

C. QUANTUM REQUIREMENTS OF QML MODELS

As discussed earlier, QuanvNNs can be broadly divided into two categories: those with trainable quantum parameters and those with fixed (nontrainable) quantum circuits. These choices entail different dependencies on quantum hardware, which can be formally categorized using the recent framework introduced in [20].

In this framework, models that require quantum hardware to compute or optimize the loss function fall under the quantum simulation (QSIM) class. This includes most variational algorithms and QML models with learnable parameters, as gradient-based training generally requires quantum hardware in the loop.

In contrast, models that utilize quantum hardware only during an initial data acquisition or encoding phase belong to the classical simulation enhanced with quantum experiments (CSIM_{QE}) class. In this case, the quantum component is used to preprocess classical data (e.g., via fixed quantum filters), and the rest of the learning pipeline, including loss computation, is performed entirely on classical hardware.

Under this classification, the quanvolutional models used in this work fall into the CSIM_{QE} class, as they use randomly initialized quantum circuits that remain fixed throughout training. By contrast, models reviewed in Section II-A, which include trainable quantum layers, would be categorized as QSIM.

Importantly, models in the QSIM class are often affected by the barren plateau phenomenon [3], in which gradients vanish exponentially with system size, making training impractical, especially on NISQ devices. By contrast, CSIM_{QE} models do not require optimization on quantum hardware and are thus considered more "NISQ-friendly." However, this does not imply that they are free of limitations. For example, even nontrainable models, such as quantum kernels [21], [22], can be affected by exponential concentration effects [23], which may limit their expressivity or robustness.

We adopt this classification in our analysis to clarify the capabilities and limitations of our models and to position our approach with respect to current and near-term quantum hardware constraints.

III. BACKGROUND

A. QUANVOLUTIONAL LAYERS

Quanvolutional layers are the fundamental component of QuanvNNs and are based on classical convolutional filters, which have transformed the field of image processing and computer vision. Similarly to their classical counterparts, they extract meaningful features from images in a locally space-invariant manner, but they also exploit the capability of quantum circuits to extract complex features that are difficult to obtain classically.

Quanvolutional layers comprise one or multiple quanvolutional filters, which perform operations on a local subsection of the input data through quantum circuits.

A filter, also called *kernel*, maps a subsection of $k \times k$ input data (pixels) $x_1, \ldots, x_{k^2} = \mathbf{x}$ to a single scalar value. The input $\mathbf{x}$ is usually referred to as a "patch." In the classical approach, this mapping is performed using classical operations, such as a scalar product between the patch values and the filter's weights, and the addition of a bias. A quantum convolutional filter acts in a similar manner, with the important difference that the mapping is performed through a quantum circuit.

More in detail, as mentioned in Section I, each filter performs the following operations.

1) *Encoding:* The classical data (i.e., pixels in the patch $\mathbf{x}$) are encoded in a quantum state.
2) *Processing:* The quantum state representing the classical data is processed through a sequence of gates.
3) *Decoding:* Classical information is extracted from the final quantum state.

The following sections describe the three phases in more detail.

B. ENCODING

In general, encoding strategies are implemented as quantum circuits, which are applied before the processing step. Their aim is to embed the classical inputs into a quantum state before further processing.

1) ROTATIONAL ENCODING

The most common approach to data encoding in quanvolutional models is through the use of rotational encoding with R_X gates [9]. The parametric gate $R_X(\theta)$ is defined as

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

and is obtained by the matrix exponential $\exp(-i\theta X)$, where X denotes one of the Pauli gates. This approach is also known as angle encoding.

In a quanvolutional filter of size $k \times k$, each pixel p_i is encoded on a different qubit. This implies that encoding k^2 pixels requires a circuit with exactly k^2 qubits. Consequently, as k increases, this approach becomes unsustainable for devices with a lower qubit count. Conversely, it may also reduce the expressivity of the filter for lower values of k.

As the matrix exponential has a period of 2π, each pixel is mapped to a rotation angle before the encoding. For example, if $p_i \in [0, 1]$, then p_i is encoded as $R_X(p_i\pi)$ applied to the ith qubit. Formally, the mapping of a patch of size $k \times k$ to the corresponding quantum state can be written as

$$|0\rangle^{\otimes n} \mapsto |\psi(\mathbf{x})\rangle = \bigotimes_{j=1}^{n} \left(\cos\left(\frac{\pi}{2}x_j\right)|0\rangle - i\sin\left(\frac{\pi}{2}x_j\right)|1\rangle\right) \tag{1}$$

with $n = k^2$.

2) THRESHOLD ENCODING

Another method of encoding classical data in quanvolutional filters is by threshold encoding. It consists in first performing a binarization of the image and then encoding pixels with value 0 with the identity gate I and pixels with value 1 with the X gate. As in the previous case, the pixel p_i is encoded in the ith qubit. This encoding process inevitably results in significant information loss due to the image binarization and can only be applied to datasets that are resilient to this procedure.

It is important to note that the threshold encoding is equivalent to the rotational encoding after performing input binarization, as $R_X(\pi)$ is equivalent to the X gate up to a global phase of $-i$.

3) HIGHER ORDER ENCODING

The rotational encoding can be enhanced with additional entangling gates, to obtain the so-called higher order encoding [12], [22]. In this encoding, after the rotational gates, there is a set of $R_{zz}(x_i x_j)$ applied to the ith and jth qubits. This encoding is more expressive, but requires additional $k^2(k^2-1)/2$ gates and a larger circuit depth.

4) OTHER NOTABLE ENCODINGS

Other encodings, which are usually not employed in quanvolutional approaches, aim to reduce the number of qubits needed to encode an image. For example, FRQI [24] can encode an image of size $k \times k$ with $2\log_2(k) + 1$ qubits, as long as k is a power of 2. However, this method requires k^4 gates. NEQR [19] is an improvement on the FRQI encoding that stores input data using the basis states instead of the amplitudes.

C. PROCESSING

Following the encoding phase, the processing section of the circuit usually consists of a randomly generated sequence of parametric and nonparametric gates. In this section, we focus on the original procedure described in [4], as it is commonly used in the literature when implementing nonlearnable QuanvNNs, and it is the method used in this work to create the processing circuits.

The parametric circuit is constructed from a set of single-qubit gates and a set of two-qubit gates.

The single-qubit gates are generated as follows: a maximum of $2k^2$ gates drawn from the set $[R_X(\theta), R_Y(\theta), R_Z(\theta), S, T, H]$. Each gate is applied to a random qubit, and θ is a random rotation parameter.

As regards the two-qubit gates, each pair of qubits q_j, q_k has a fixed probability (usually $p = 0.15$) of having a gate applied to them. The gate is randomly selected from the set $[CNot, Swap, SqrtSwap]$. The qubit selection is inspired by random graph models [25], where each qubit is treated as a vertex, and a two-qubit gate between them is treated as an edge.

Finally, all the generated gates (single- and two-qubit) are randomly shuffled, obtaining the order of the gates for the final circuit $\mathcal{U}$.

D. DECODING

Finally, each quantum state must be translated into a classical scalar value, in order to construct a new input for the following layer of the model. To achieve this, it is first necessary to measure each quantum state. Subsequently, the distribution obtained from the measurement can be converted into a real number. This step can be performed in several ways, e.g., the number of qubits in the $|1\rangle$ state in the most measured state can be counted [4]. A different approach is the one used in [12], where the authors obtain the expectation value for each observable in the circuit. Therefore, the output of the circuit is a vector instead of a single number, i.e., an output channel is generated for each qubit in the circuit.

E. EXPRESSIBILITY OF A QUANTUM CIRCUIT

Expressibility is one of the most significant descriptors of a parametric quantum circuit. It can be defined as the ability of the circuit to uniformly cover the Hilbert space of the underlying quantum system (i.e., the circuit's ability to explore the Bloch sphere in the case of a single qubit). Moreover, researchers have shown a strong positive correlation between the expressibility of a quantum circuit used in a variational quantum classifier and its performance [16].

The expressibility index has first been proposed in [26]. It is calculated by comparing the fidelities distribution of states

obtained by a circuit $\mathcal{U}$ to the fidelities distribution of random states of the system, which corresponds to the Haar random states ensemble.

In order to compute the expressibility, the authors first approximate the former distribution by randomly sampling two sets of parameters $\theta_1, \theta_2 \in \Theta$ of the parametric quantum circuit $\mathcal{U}$. They then compute the fidelity $|\langle \mathcal{U}(\theta_1)|\mathcal{U}(\theta_2)\rangle|^2$ between the states obtained with the corresponding sets of parameters.

Subsequently, they compute the Kullback–Leibler divergence between the distribution $\tilde{P}_{\mathcal{U}}(F;\Theta)$ and the distribution of random states $P_{\text{Haar}}(F)$, which is known to be equal to $(N-1)(1-F)^{N-2}$, where N is the dimension of the quantum system [27], obtaining

$$\text{Expr}\left(\mathcal{U}(\boldsymbol{\theta})_{\theta\in\boldsymbol{\Theta}}\right) = D_{\text{KL}}\left(\tilde{P}_{\mathcal{U}}(F;\Theta)\middle\|P_{\text{Haar}}(F)\right). \quad (2)$$

The formula to compute D_{KL} of two continuous random variables P and Q is

$$D_{\text{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\log\left(\frac{p(x)}{q(x)}\right)dx \quad (3)$$

where p and q denote the probability densities of P and Q, respectively.

If the value of D_{KL} (and therefore Expr) is close to zero, then the two distributions are similar (i.e., in our case, the parametric quantum circuit $\mathcal{U}$ is very expressive).

It is important to note that, in general, expressibility is computed for variational circuits, while the circuits considered in this work have randomly generated or feature-dependent parameters. In this context, it represents the ability of a circuit to extract diverse features from the input data.

IV. PROPOSED METHOD

A. DATA QUANTIZATION

To enhance computational efficiency during the preprocessing stage, two techniques are employed: first, the input data are quantized, and then, quantum circuit outputs for each unique patch are memoized. Previous works utilized a binary image quantization and a lookup table (memoization) to expedite dataset processing. In general, binary quantization significantly reduces data fidelity, with the potential for complete loss of information (see Fig. 2). Therefore, a higher number of quantization levels are employed to preserve as much information as possible while maintaining computational practicality. The proposed quantization approach extends binary quantization by introducing a quantization level, denoted as N.

The formula used to quantize a pixel is

$$q(x) = \frac{\lfloor x\cdot N\rfloor}{N-1} \quad (4)$$

where $x \in [0, 1)$ is the original pixel intensity. In other terms, when an image is quantized to N levels, we first extract the N points $\{0, \frac{1}{N-1}, \frac{2}{N-1}, \ldots, 1\}$ from the interval [0,1]. Then,

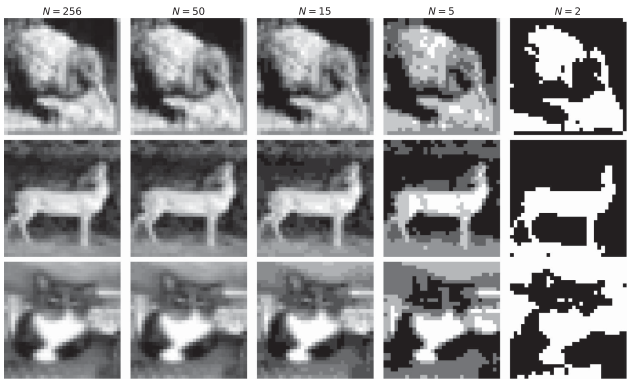

FIGURE 2. Visualization of quantization for different values of *N*, on three images from the CIFAR10 dataset (belonging to the classes bird, horse, and cat). Quantization is applied after converting to gray scale.

for each image I, each pixel value is mapped to the closest point in the set, obtaining a quantized image $q(I)$.

The memoization technique is implemented by constructing a lookup table that links each quantized input patch to the output computed by the circuit. The table is filled-in dynamically while preprocessing the dataset. This process ensures that unnecessary computations for absent patches are avoided, as only the patches actually encountered in the dataset are processed. This methodology allows one to balance the tradeoff between detail retention (preserving information) and computational load (efficient processing). A higher N value offers finer details at the expense of expanding the memoization table and needing more quantum circuit executions. Conversely, a lower N value simplifies the process and reduces computational demands, but this is achieved at the cost of losing input information. Fig. 2 displays some examples of the effect of quantization for different values of N.

To evaluate the information loss caused by using different quantization levels, we can calculate the mean squared error (MSE) between the original image and its quantized version. An upper limit can be estimated by considering that (4) maps each value $x \in [0, 1]$ to a point $q(x)$ such that $|x - q(x)| \leq \frac{1}{2(N-1)}$; therefore, the error is

$$\text{MSE}\left(I; q(I)\right) \leq \frac{1}{4\left(N-1\right)^2}. \quad (5)$$

Nevertheless, the actual loss incurred during the quantization process ultimately depends on the specific dataset.

Without memoization, preprocessing a dataset would require a number of quantum circuit executions equal to the product of the number of input patches and the number of filters. Even for relatively modest datasets, this approach is impractical on current quantum hardware. Previous methods relied on binarization (i.e. quantization for $N = 2$) of the dataset to drastically reduce the amount of patches to compute (for example, the upper limit on the number of patches

IEEE Transactions on Quantum Engineering

of size 3×3 is $2^9 = 512$). However, larger values of N might still provide a significant reduction in the number of unique patches, depending on the dataset.

Since both MSE and the number of unique patches for a quantization level are computationally easy to obtain, these value can be used as indices to balance the information loss with the required processing time, without defaulting to an aggressive data binarization.

B. INTEGRATED ENCODING

The proposed method is based on the use of classical data as a rotation angle for multiqubit entangling gates, with the objective of directly encoding data into the processing section of the circuit. The rationale behind this approach is that it removes the dependence between the size of the kernel and the number of qubits required, thus enabling their selection independently. This approach is similar to some applications of quantum kernels [21], where classical data are directly encoded in a quantum state without strict constraints on the number of qubits.

The parametric gates used in our model are obtained from the set of Pauli gates $P = \{\sigma_I, \sigma_X, \sigma_Y, \sigma_Z\} = \{I, X, Y, Z\}$. Given two elements $\sigma_1, \sigma_2 \in P$, we can construct the parametric gate $G(\theta)$ as $\exp(-i\theta\sigma_1 \otimes \sigma_2)$, where exp is the matrix exponential and $\otimes$ is the tensor product. As an example, consider $\sigma_1 = \sigma_2 = X$. Then, $\exp(-i\theta X \otimes X) =$

$$\begin{bmatrix} \cos(\theta) & 0 & 0 & -i\sin(\theta) \\ 0 & \cos(\theta) & -i\sin(\theta) & 0 \\ 0 & -i\sin(\theta) & \cos(\theta) & 0 \\ -i\sin(\theta) & 0 & 0 & \cos(\theta) \end{bmatrix}.$$

Consider a filter of size $k \times k$, and a quantum circuit with n qubits. A gate G in the quantum circuit is constructed as follows: a random feature $x_i \in \{x_1, x_2, \ldots, x_{k^2}\}$ (corresponding to the intensity of a pixel in the patch $k \times k$) is selected, along with two gates $\sigma_1, \sigma_2 \in P$, two distinct qubits $q_j, q_k \in \{q_0, q_1, \ldots, q_{n-1}\}$, and a mapping function $\alpha : [0, 1] \to [0, 2\pi]$. Then, the gate G is defined as

$$G(x_i) = \exp\left(-i\alpha(x_i) \cdot \sigma_1 \otimes \sigma_2\right)[q_j, q_k] \tag{6}$$

where the notation $[q_j, q_k]$ indicates that the gate is applied on qubits q_j and q_k. The pair of qubits is randomly selected among all possible pairs. However, when working on a real quantum device, it is possible to base the selection on the connectivity of the device. Finally, the mapping function $\alpha(\cdot)$ can be selected to be any mapping from the input space (i.e., pixel intensities) to rotation angles. In the simplest case, it can be chosen as $\alpha(x) = x\pi$. Nevertheless, it is possible to select alternative functions, such as random or learnable linear transformations, or standard activation functions such as the sigmoid function.

The complete circuit can be written as a concatenation of L gates $\prod_{l=1}^{L} G_l(x_{j_l})$.

To prevent information loss, we force the circuit generation to use each feature x_i in the patch at least once. The number of gates L can be selected according to hardware limitations and desired properties, as long as $L \geq k^2$ to encode all the features. However, we expect the optimal number of gates L to scale polynomially in both the number of features and the number of qubits.

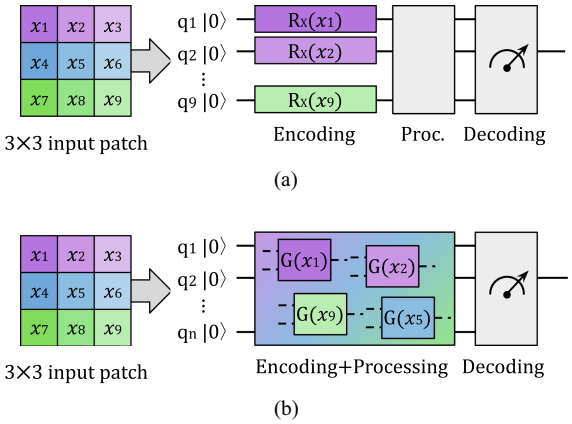

FIGURE 3. Graphical representation of the encoding methods compared in this work. (a) Rotational encoding. (b) Integrated encoding.

While the aforementioned description focuses on 2-D filters, the same methodology can be extended to 3-D filters for multichannel images by selecting $k \times k \times c$ input features, where c is the number of channels.

1) COMPARISON WITH ROTATIONAL ENCODING

One of the limitations of rotational encoding is its lack of flexibility. In particular, a filter of size $k \times k$ requires exactly k^2 qubits and an average number of basic gates equal to $k^2 + pk^4$, where the former term depends on the single-qubit gates, while the latter depends on the two-qubit connections between the k^2 qubits (see Section III-C). This results in circuits that may be excessively complex (in both depth and number of qubits) when the size of the filter k is high or insufficiently expressive when k is low.

In contrast, our proposed model allows for significant flexibility in both the number of qubits and the number of operations L.

Fig. 3 shows a simplified comparison of the structure of a processing circuit using (a) rotational encoding and (b) the proposed integrated encoding.

V. EXPERIMENTAL DESIGN

The experiments are conducted on binary and multiclass classification problems across two different datasets. The aim of the experiments is to understand the performance of the proposed integrated quanvolutional model for different mapping functions $\alpha(\cdot)$, compared to the standard rotational encoding approach. Each setting is tested for different kernel sizes and is finally compared to a classical CNN.

A. DATASETS

To test our proposed model, we selected two image classification datasets from different fields.

1) *MiraBest* [14] comprises images of galaxies, classified according to the Fanaroff–Riley morphology into three macroclasses: FR-I, FR-II, and Hybrid. We use Version 1[2] of the dataset, which consists of only the samples labeled as Confident (as opposed to Uncertain) and discards the Hybrid class, which contains 19 Confident samples only. The resulting dataset comprises 770 samples, of which 339 belong to the FR-I class and 431 to the FR-II class. Each sample is normalized and downscaled to a size of 30×30.
2) *LArTPC* [15] is a dataset consisting of realistic simulations of particle activities. Each particle belongs to a class in the following categories: e^-, μ^+, p, γ, π_0, π^+, and K^+. Each class contains 100 samples. Each sample is represented as a 2-D matrix, where one axis is the position in the sensing wire, the other axis is the time sampling tick, and the "pixel intensity" is the energy loss at the corresponding position and time. The size of each matrix is 480×600. Samples are first normalized using a MinMax Scaler to ensure that each pixel is in [0, 1] and then downscaled to a size of 30×30, following the original paper.

The datasets used in this work were selected based on three main considerations. First, their relatively small size ensures practical processing times in quantum circuit simulations, which remain computationally expensive. Second, their limited number of training samples provides a natural testbed for evaluating the behavior of quantum models in low-resource scenarios. This is particularly relevant given recent findings suggesting that quantum models may generalize better than classical ones when data are scarce [28]. Third, and importantly, both datasets originate from scientific domains deeply rooted in quantum phenomena, namely, the classification of astrophysical radio emissions and the identification of subatomic particle interactions. This alignment with quantum-native processes offers a conceptually meaningful and coherent setting in which to evaluate QML pipelines. Despite their size, both datasets present substantial classification challenges due to high intraclass variability and subtle interclass differences, further supporting their suitability for this study.

As regards the MiraBest dataset, we used the fixed train–test split provided by the original authors (90%–10%), which results in 693 train images and 77 test images. For the LArTPC dataset, the data were randomly divided into a training set and a test set (85% –15%) at each experimental run.

In addition, we employ the following datasets to test the effects of data quantization and analyze the trade-off between information loss and reduction in circuit executions.

1) *MNIST* [29] comprises images of handwritten digits for ten-class digit classification. The dataset contains 70 000 images of size 28×28.

[2] https://zenodo.org/records/4288837

TABLE 1. Number of Samples and Sample Size for Four Image Classification Datasets: MNIST, CIFAR10, MiraBest, and LArTPC

Dataset	Samples	Image Size	Number of 3×3 patches
MNIST	70,000	28×28	47,320,000
CIFAR10	60,000	32×32	54,000,000
MiraBest	770	30×30	603,680
LArTPC	700	30×30	548,800

Patches of size 3×3 are used for illustrative purposes.

2) *CIFAR10* [30] is a dataset for general image classification, containing 60 000 images of size 32×32 belonging to ten classes of vehicles and animals.

Table 1 summarizes the basic information about the datasets, such as the number of samples, image size, and the total number of unique 3×3 patches.

B. TESTED MODELS

We compare our proposed quanvolutional model with integrated encoding (QNN-Int) with a classical CNN model and a quanvolutional network using rotational encoding (QNN-Rot). In addition, we test three different mapping functions $\alpha(\cdot)$ to map pixel intensities to rotations angles in QNN-Int. Finally, for both quanvolutional models, multiple kernel sizes are tested to determine their effect on the model's performance. Each quanvolutional layer employs eight circuits. This number was chosen to balance model expressivity with computational cost, based on preliminary experiments that showed diminishing returns in accuracy beyond this point, consistent with findings reported in [4].

Following are the implementation details of the models. Fig. 4 shows a schema of the architecture.

1) *CNN—a classical convolutional neural network:* The structure of the model contains one convolutional layer and two fully connected layers. The convolutional layer consists of 16 output channels, has a filter size of 3×3 with no padding, and is followed by a ReLU, and a Max Pooling layer of size 2×2. The first fully connected layer has 32 output features, while the second one, which is also the output layer, has a number of outputs equal to 7 or 2, depending on the task. Each fully connected layer is followed by a ReLU activation function. The convolutional layer and the first fully connected layer are followed by a Dropout layer, with probability of 0.2, to prevent overfitting.
2) *QNN-Rot—quanvolutional model with rotational encoding:* The model consists in the same architecture described above, with a quanvolutional layer stacked on top. The number of output channels on the quanvolutional layer is 8, and the first Conv layer is modified is take as input 8 channels instead of one. Each filter is padded to have the same input and output dimensions. The tested filter sizes are: 2×2, 3×3, and 4×4, with a qubit requirement n of 4, 9, and 16. Larger filter sizes were not tested due to the high resource demands

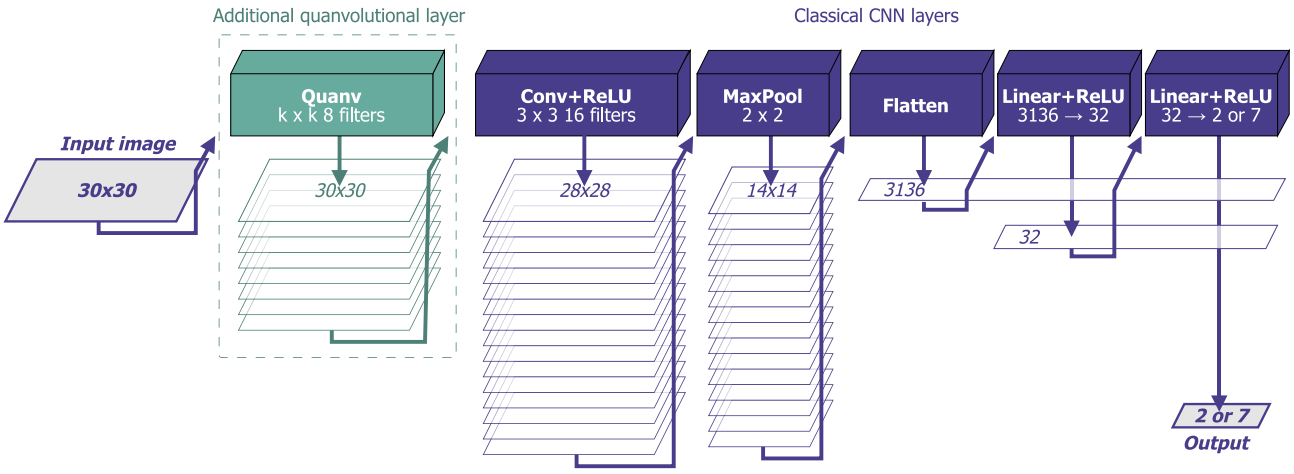

FIGURE 4. Schema of the CNN and QNN-based architectures. The QNN-based architectures are obtained by adding a quanvolutional layer (in green) before the classical CNN layers (in blue).

associated with their simulation, which was unfeasible on our device. The processing circuit follows the standard implementation described in Section III-C. The connection probability used for the circuit generation is set to 0.15.

3) *QNN-*INT*—quanvolutional model using the proposed integrated circuit:* The structure of the model is the same as the one described for the QNN with the rotational encoding, with the quanvolutional layer on top of the CNN. The layer comprises eight channels, and each filter is padded so that the output of the layer has the same dimensions of the input.
The number of qubits is set to $n = 4$ to balance the expressivity of the model with the resources required for its implementation. The tested filter sizes are 2×2, 3×3, 4×4, and 5×5.
Regarding the mapping function $\alpha(\cdot)$, we considered three options representing baseline linear transformations with increasing parameter complexity.
   1) SIMPLE: $x \mapsto x\pi$; it directly maps input values to angles without further adjustments, a common strategy in quantum data encoding.
   2) RNDMUL: $x \mapsto 2\beta x\pi$; it introduces a random scaling factor (β) to increase diversity.
   3) RNDLIN: $x \mapsto (\beta x + \sigma)\pi$; it generalizes further with both scaling (β) and bias (σ), forming the simplest complete linear transformation.

   Here, x is the pixel value normalized in $[0, 1]$, and β and σ are random parameters drawn uniformly from $[0, 1]$, independently for each gate. The SIMPLE function directly maps input feature to angles, mirroring standard rotational encoding. RNDMUL and RNDLIN extend this by adding an increasing number of random parameters through basic linear operations, offering a progression in flexibility.
   As discussed in Section IV-B, the number of gates L should be at least equal to k^2 to encode all input features. We set $L = 2k^2$ and force each feature to be encoded at least once in the final circuit.

C. DECODING

To obtain the output of the quanvolutional filter, we need a way to decode the quantum state through a measurement. In this work, we choose to measure the output state $\mathcal{U}(\mathbf{x})|\psi_0\rangle$ with the projector $\mathcal{M} = Z^{\otimes n}$. The expectation value is obtained as

$$p = \langle\psi_0|\mathcal{U}(\mathbf{x})^\dagger \mathcal{M}\mathcal{U}(\mathbf{x})|\psi_0\rangle. \tag{7}$$

To translate the output measurement to a scalar value, we compute the average number of qubits that were measured in the $|1\rangle$ state after repeating the measurement for a fixed number of times. This method was chosen as it is more resilient to the stochasticity of the measurements than the one based only on the most common measured output [4]. As an example, consider a circuit that for a given input patch $\mathbf{x}$ returns an equal superposition of the states $|0\rangle^{\otimes n}$ and $|1\rangle^{\otimes n}$. If the filter output depends only on the most common state measured, then the output has a 50% chance of being 0 or 1, while by taking into account all the measured states we have an output of 0.5.

D. IMPLEMENTATION DETAILS

All models are trained using the ADAM optimizer with a learning rate of 0.0003, with a batch size of 16. We use the negative log-likelihood loss on log-softmax outputs. The training process employs early stopping with a patience of 10, i.e., the training halts if the test loss fails to improve over ten consecutive epochs. For the QNN-INT-RNDLIN model, the patience is increased to 100, as the training loss took significantly more epochs to decrease. Each training is repeated ten times with different random seeds.

The model architectures, training procedures, and testing methodologies are implemented in PyTorch. Quantum operations are performed with noiseless state vector simulation

with Qiskit 1.1. The decoding of the circuits is performed by a standard measurement in the computational basis for each qubit and by computing the average (i.e., the final output is the average number of qubits in the state $|1\rangle$) for a fixed number of samples equal to 1000. In general, by increasing the number of qubits in the system, it is expected that (exponentially) more measurements are required to obtain a good estimation of the quantum state. However, since the number of qubits used in the experiments is small, we did not observe any significant difference when increasing the number of shots. All experiments are executed on an Intel Xeon W-2123 machine with 48-GB RAM.

A small-scale noisy evaluation (see Appendix C) is also carried out using the FakeTorino backend to simulate realistic hardware noise; this experiment only involves the best performing configuration from the main experiments.

E. EXPRESSIBILITY MEASUREMENT

To compute an approximation of the expressibility Expr of a quanvolutional circuit given in formula (2), we compute 2^{10} fidelities of the circuit by creating pairs of random input vectors. We then calculate the discretized version of Expr as follows:

$$\sum_{i\in\text{bins}} P(i)\log\left(\frac{P(i)}{Q(i)+\varepsilon}\right) \tag{8}$$

where bins are obtained by dividing the interval [0,1] in 50 equal-sized intervals, $P(i)$ is the number of fidelities in the ith bin, and $Q(i)$ is obtained from the distribution of P_{Haar}. The additive constant $\varepsilon = 10^{-16}$ is used for numerical stability in the computation.

All the randomly initialized nonlearnable parameters of the circuits (e.g., θ in the QNN-Rot, and σ and β in QNN-Int) are maintained fixed while measuring the 2^{10} fidelities.

The expressibility measurement described earlier is repeated ten times for differently initialized circuits.

VI. EXPERIMENTAL RESULTS

In the following, we present an analysis of the effect of data quantization on the datasets. Then, we report the results of the classification experiments conducted on MiraBest and LArTPC. Finally, we perform an expressibility analysis for QNN-Int and QNN-Rot.

A. DATA QUANTIZATION

We perform a preliminary analysis of the impact of data quantization, restricted to 3×3 patches on the four datasets: MNIST, CIFAR10, MiraBest, and LArTPC. We focus on two metrics: information loss and reduction in number of circuit execution. The former is calculated as the MSE between the original and quantized images. The latter is calculated by comparing the total number of 3×3 patches in the whole dataset and the number of unique 3×3 patches obtained after quantization.

Fig. 5 shows the trend of the information loss (MSE, y-axis) depending on the quantization level (x-axis) for all

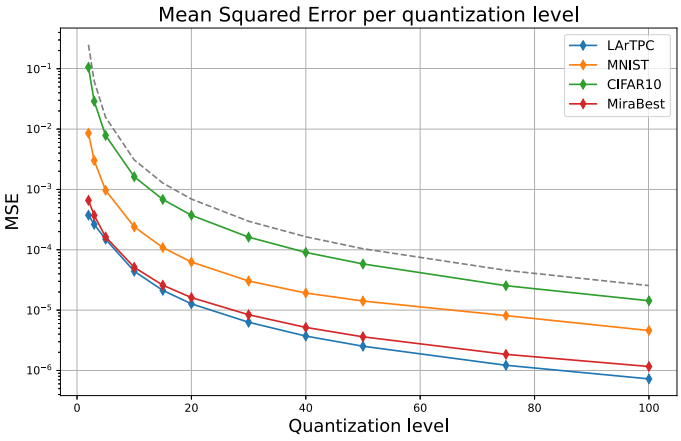

FIGURE 5. Average MSE on the considered datasets after applying quantization to N levels, shown on a logarithmic scale. The dashed line is the maximal MSE given by (5).

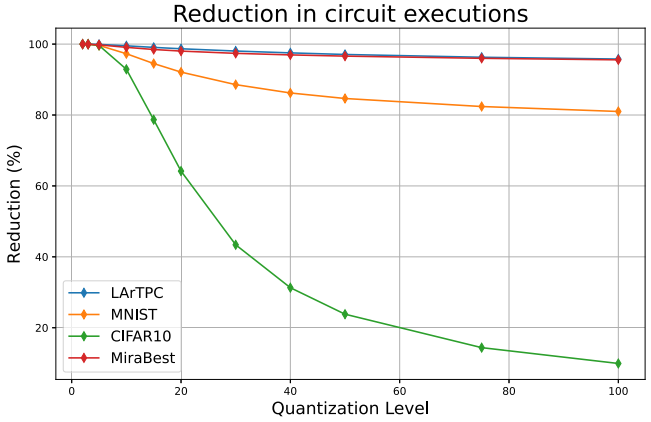

FIGURE 6. Reduction in the number of circuit executions after applying quantization (N levels) and memoization to the considered datasets with patch size 3 × 3.

datasets. The theoretical maximum MSE between the original and quantized images for each quantization level N, given by (5), is shown as a dashed line. The actual average MSE calculated on the datasets is shown as solid lines. We can see that the actual MSE on the datasets can be significantly lower than the upper bound (e.g., for LArTPC and MiraBest). This largely depends on the variability of pixel intensities in the original dataset. For example, CIFAR10 contains real-life images of animals, vehicles, and objects (see Fig. 2), resulting in a high variability and an MSE close to the upper bound. On the other hand, the images in LArTPC and MiraBest contain large black backgrounds with (relatively) small mostly white objects, which lead to lower error rates during quantization. The acceptability of information loss is contingent upon the characteristics of the dataset in question. For instance, $N = 2$ is an acceptable quantization level for MNIST, whose images remain recognizable after the process, but not for CIFAR10. In general, we expect that levels of $N \ll 20$ may result in a loss of information that negates any potential quantum

TABLE 2. Reduction in the Number of Circuit Executions After Applying Quantization (N Levels) and Memoization to the Datasets LArTPC, MiraBest, and MNIST With Patch Size 3 × 3

Quantization Level	5	10	15	20	30	**50**	100
LArTPC	99.91	99.51	99.07	98.68	98.04	**97.07**	95.79
MNIST	99.64	97.29	94.51	92.10	88.57	84.66	80.99
CIFAR10	99.62	92.90	78.68	64.18	43.40	23.82	9.93
MiraBest	99.79	99.08	98.48	98.03	97.39	**96.61**	95.56

The level selected for the experiments in this work is in bold.

TABLE 3. Test Accuracy on MiraBest (Left) and LArTPC (Right) (Average and Standard Deviation Over Five Seeds), With and Without Quantization

	MiraBest		LArTPC	
Model	$N = 50$	$N = \infty$	$N = 50$	$N = \infty$
QNN-Rot	77.14 ± 2.53	77.40 ± 1.97	53.00 ± 1.85	52.71 ± 2.22
QNN-Int-Simple	79.22 ± 3.05	81.56 ± 3.83	57.86 ± 0.87	59.57 ± 2.29
QNN-Int-RndMul	80.26 ± 3.23	81.56 ± 5.69	58.14 ± 3.73	57.86 ± 2.86
QNN-Int-RndLin	60.26 ± 9.87	60.00 ± 9.29	46.29 ± 19.48	39.14 ± 23.35

advantage that could be obtained through quanvolutional approaches.

Fig. 6 reports the percentage of reduction in circuit executions (y-axis) given the quantization level (x-axis) for all datasets. The plot shows that quantizing to $N = 10$ levels reduces the number circuit executions by 92% for all datasets. The reduction reaches 99% for MiraBest and LArTPC, potentially allowing significant savings in terms of resource utilization. As the number of quantization levels increases, the effect of quantization remains significant for several datasets: $N = 100$ leads to a 95% reduction for MiraBest and LArTPC, and an 80% reduction for MNIST. On the other hand, we observe that for CIFAR10, the amount of reduction in circuit executions becomes minor for $N \gg 20$, reaching 43%, 24%, and 10% for $N = 30, 50$, and 100 respectively.

This confirms the previous observations on the information loss, showing that the efficacy of this method ultimately depends on the characteristics of the dataset.

Given these results, the MiraBest and LArTPC datasets used in the following experiments are preprocessed with $N = 50$ quantization levels. The value of N is selected to obtain a high computational speed-up in the quanvolutional layer application (respectively, 96% and 97% reduction in circuit executions; see Table 2) with minimal information loss (MSE$< 10^{-5}$; see Fig. 5).

B. COMPARISON WITH NONQUANTIZED IMAGES

To further assess the impact of using $N = 50$ quantization levels, we compare the performance and computational requirements of all models with and without quantization (the latter referred to as $N = \infty$). As before, this preliminary analysis is restricted to models with $k = 3$ using the MiraBest and LArTPC datasets.

1) PERFORMANCE

Table 3 reports the average classification accuracy of all models on the two datasets, both with and without quantization. Results are averaged over five runs.

On MiraBest, QNN-Int-Simple and QNN-Int-RndMul show a modest performance drop with quantization (about 2–3 percentage points), while other models exhibit no notable differences. On LArTPC, QNN-Int-Simple again incurs a performance loss of approximately 3 percentage points, QNN-Int-RndLin shows a performance drop of 5 points (but with a large variability between runs), with the remaining models showing no significant degradation.

These results confirm that the low information loss observed a priori (see Fig. 5) translates into only minor losses in classification performance.

2) TIME REQUIREMENTS

The time required to preprocess the dataset is linear in the number of patches processed. With the proposed memoization strategy, the number of unique patches, and therefore the number of quantum circuit executions, is strongly dependent on the quantization level.

Note that memoization can still be applied even without quantization, since the input data are inherently discretized, and certain patches (e.g., all-black regions in LArTPC corresponding to zero energy) are repeated. For comparison, we also estimate the time required to preprocess each dataset without using memoization. Specifically, this is calculated as the time to evaluate a single patch multiplied by the total number of patches in the dataset. Results are reported in Table 4.

On MiraBest [see Table 4(a)], removing quantization increases preprocessing time by a factor of 6 on average (range: 5.6–6.1 depending on the model). Removing memoization altogether results in a much larger slowdown of 29× (range: 28.0–30.4). In total, using $N = 50$ quantization levels and memoization, compared to no memoization at all, yields an average computational speed-up of 96.56%, closely matching the 96.61% reduction in circuit executions reported in Table 2.

On LArTPC [see Table 4(b)], the impact of quantization is slightly smaller: using $N = \infty$ (no quantization) results in an average slowdown of 3.2× (range: 2.9–3.5). Disabling memoization leads to a slowdown of 34× (range: 30.9–37.0).

TABLE 4. Time Required (In Minutes) to Preprocess (a) MiraBest and (b) LArTPC Using $N = 50$ Quantization Levels, No Quantization, and No Memoization

Model	With memoization $N = 50$	With memoization $N = \infty$	No memoization
QNN-Rot	62.70	349.71 (×5.6)	1755.70 (×28.0)
QNN-Int-Simple	53.29	298.42 (×5.6)	1498.18 (×28.1)
QNN-Int-RndMul	52.22	309.34 (×5.9)	1553.02 (×29.7)
QNN-Int-RndLin	53.13	322.18 (×6.1)	1617.47 (×30.4)

(a) MiraBest

Model	With memoization $N = 50$	With memoization $N = \infty$	No memoization
QNN-Rot	49.82	150.43 (×3.0)	1580.94 (×31.7)
QNN-Int-Simple	37.54	132.23 (×3.5)	1389.71 (×37.0)
QNN-Int-RndMul	37.19	129.61 (×3.5)	1362.08 (×36.6)
QNN-Int-RndLin	43.46	127.99 (×2.9)	1345.07 (×30.9)

(b) LArTPC

Numbers in parentheses show the time increase compared to the quantized model. All values are calculated for eight quanvolutional filters with kernel size 3 × 3. The values for no memoization are estimated.

In this case, using $N = 50$ with memoization yields a 97.05% speed-up, again matching the 97.07% reduction in circuit executions shown in Table 2.

C. CLASSIFICATION PERFORMANCE

1) RESULTS ON MIRABEST

Table 5(a) reports the average classification accuracy of all models tested on the MiraBest dataset (two classes).

The average accuracy of the classical CNN model is 71.039%.

QNN-Rot surpasses the performance of the CNN for $k = 2, 3$, reaching up to 77.273% accuracy. Its performance degrades as k increases, and becomes unstable with $k = 4$, with 10 points of standard deviation.

QNN-Int-Simple and RndMul significantly outperform both the baseline CNN and QNN-Rot for $k = 2, 3, 4$, and RndMul also matches the performance of the CNN with $k = 5$. QNN-Int-RndMul reaches the best overall performance with 80.779% accuracy at $k = 3$.

The QNN-Int-RndLin model exhibits suboptimal performance for all values of k, as its performance remains 10–20 points lower than the classical CNN.

In general, lower values of k tend to perform better, while there is a decline in performance as k increases to 4 and 5.

2) RESULTS ON LARTPC

Table 5(b) reports the average classification accuracy of all models tested on the LArTPC dataset (seven classes).

The average accuracy of the classical CNN model is 56.789%.

The performance of the QNN-Rot model is lower than the CNN for all values of k. Its performance increases with k, and it reaches a maximum of 52.929% for $k = 4$.

QNN-Int-Simple and QNN-Int-RndMul have similar patterns in performance. For $k = 2$, their accuracy is lower than the CNN, but still higher than the best performance of QNN-Rot. For $k = 3, 4, 5$, both models surpass the performance of the CNN, reaching their best performance at $k = 4$ (58.500 for Simple and 57.786 for RndMul).

QNN-Int-RndLin never surpasses the baseline CNN performance, reaching a maximum of 54.071% accuracy with $k = 2$, and showing a large standard deviation for $k = 3, 4, 5$ (up to 18 points). This shows that RndLin is highly unstable on this dataset.

In contrast to the MiraBest dataset, larger filters obtain better results, but the best performing models exhibit a smaller advantage over the classical approach. The difference in performance trend appears to be related to the distribution and diversity of $k \times k$ patches. Specifically, MiraBest exhibits low patch diversity, with most patches concentrated in a compact region or near-zero values even for higher values of k. On the other hand, the LArTPC dataset exhibits greater patch diversity with larger patches, providing additional information for the classification models. Appendix A contains more details about this analysis.

D. EXPRESSIBILITY ANALYSIS

Fig. 7 shows a comparison of the expressibility of the proposed QNN-Int model with QNN-Rot using different kernel sizes. To increase the readability of the plots, we show the value of Expr′, computed as $\text{Expr}' = -\ln(\text{Expr})$, so that higher values correspond to an increased expressivity.

When computing the expressibility of the proposed QNN-Int circuit with a fixed number of qubits and features (Fig. 7, upper row), we observe that increasing the number of gates L (moving right on the x-axis) leads to increasingly more expressive circuits (increased Expr′). This implies that it is possible to select a priori the value of L in order to obtain a desired value of expressibility.

Increasing the kernel size k leads to a higher increase in expressibility with a lower number of gates. For example, QNN-Int-Simple reaches $\text{Expr}' = 2$ with 25 gates when $k = 2$, 14 gates with $k = 3$, and 12 gates with $k = 4$.

When comparing the three different mapping functions α, we observe that Simple and RndMul display a similar trend, while RndLin has significantly lower expressibility for all kernel sizes. Therefore, we expect RndLin to reach a lower classification accuracy compared to the other two functions.

As regards QNN-Rot, the x-axis of the plots in the lower row of Fig. 7 reports the probability p instead of the number of gates. This is because the number of gates in a circuit with rotational encoding depends on p and k, and it averages at $k^2 + pk^2(k^2 - 1)$. One can select different values of the connection probability p to obtain circuits with different length. We observe that QNN-Rot circuits are generally less expressive than QNN-Int. In addition, their expressibility does not depend on the number of operations involved. For every value of p, there is no statistically significant difference in the value of Expr′, which only increases with the kernel size k.

IEEE Transactions on Quantum Engineering

TABLE 5. Classification Accuracy for the (a) MiraBest and (b) LArTPC Datasets

Model	$k=2$	$k=3$	$k=4$	$k=5$
QNN-Rot	**77.273** ± 2.740	**75.584** ± 2.309	67.922 ± 9.993	–
QNN-Int-Simple	**79.870** ± 2.126	**79.870** ± 2.548	**74.156** ± 6.859	69.351 ± 6.118
QNN-Int-RndMul	**80.130** ± 2.098	**80.779** ± 2.158	**73.247** ± 7.731	71.039 ± 5.896
QNN-Int-RndLin	63.636 ± 9.719	61.492 ± 8.771	57.792 ± 3.907	58.052 ± 6.623
CNN	71.039 ± 12.632			

(a) MiraBest

Model	$k=2$	$k=3$	$k=4$	$k=5$
QNN-Rot	51.571 ± 3.270	52.357 ± 3.900	52.929 ± 4.354	–
QNN-Int-Simple	54.143 ± 3.044	**56.857** ± 3.067	**58.500** ± 2.246	**58.429** ± 2.299
QNN-Int-RndMul	54.071 ± 2.434	**57.643** ± 3.147	**57.786** ± 2.106	**57.786** ± 2.565
QNN-Int-RndLin	54.071 ± 3.572	45.786 ± 16.368	49.571 ± 18.237	51.286 ± 13.328
CNN	56.786 ± 9.182			

(b) LArTPC

The reported results are the average and standard deviation over ten runs. Numbers in bold correspond to values higher than the baseline CNN accuracy.

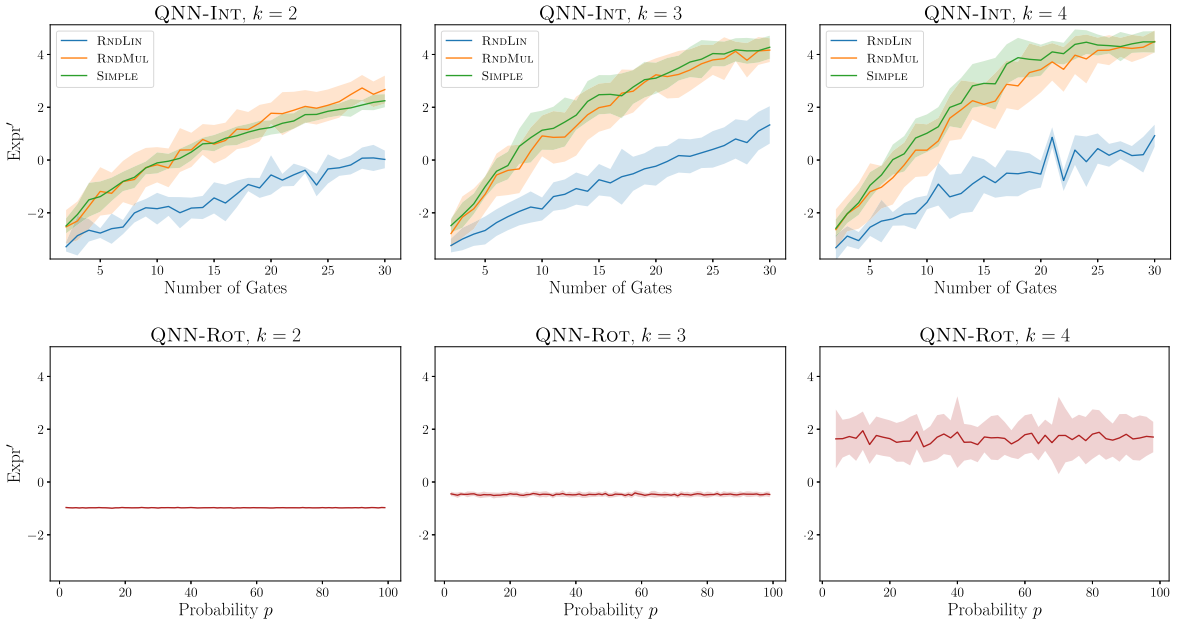

FIGURE 7. Expressibility of parametric quantum circuits, computed as described in formula (2) by uniformly sampling parameters in [0,1]. For readability purposes, we show the value Expr′, computed as Expr′ = − ln(Expr). For the model QNN-Int, the number of qubits is set to 4. All plots report the average expressivity (solid line) and its standard deviation (shadowed area) over ten random circuits.

These observations are in line with the experimental results, which show that QNN-Int-Simple and QNN-Int-RndMul tend to outperform QNN-Rot in classification tasks. However, additional experiments presented in Appendix B show that expressibility alone is not a strong predictor of model performance for quanvolutional circuits in our setting.

E. DISCUSSION

Overall, the results of the classification experiments on both datasets show that the proposed QNN-Int model reliably surpasses the performance of the QNN-Rot model for all kernel sizes and was always able to surpass the performance of a classical CNN.

In the QNN-Int model, the function chosen to map pixel intensities to rotation angles plays an important role and can lead to very different performances. Among the tested functions, Simple and RndMul performed the best, while RndLin often reached lower performances even compared to QNN-Rot. In addition, RndLin also exhibited the highest performance variance. As the most complex mapping function, it involves a large number of randomly initialized parameters, which may contribute to this variability. This

suggests that higher parameter count does not improve generalization when using nontrainable quanvolutional filters, and may even hinder it.

Increasing the kernel size k led to different results in the two datasets. On MiraBest, the performance of the models seems to decrease with k, while on LArTPC, the best results are achieved for higher values of k. This contrast appears to be related to the distribution of $k \times k$ patches. Specifically, in MiraBest, higher values of k result in a large number of patches with near-zero values, while the LArTPC dataset exhibits greater patch diversity. See Appendix A for a more detailed analysis.

The best performing model (QNN-INT-RNDMUL, $k = 3$ on MiraBest) was also evaluated under simulated realistic noise (see Appendix C). Its accuracy remained comparable to the noiseless case (79.22% with noise versus 80.52% without), indicating that the proposed approach can operate effectively under realistic noise conditions.

Finally, quantization significantly reduced preprocessing time with negligible performance loss. For datasets containing many repeated patches, memoization remains effective even without quantization, as digital images are already quantized (e.g., to 255 levels), enabling substantial speed-ups in practice.

VII. CONCLUSION

In this work, we presented a new quanvolutional model and preprocessing pipeline to make data quantization, encoding, and processing more efficient on NISQ devices.

The proposed flexible quantization approach enabled a significant reduction in the number of quantum circuit executions required to process the datasets considered in this work. In particular, we obtained a reduction of over 95% circuit executions when using 3×3 kernels, while losing a negligible amount of information. This technique has the potential to be highly beneficial for quanvolutional approaches applied to tasks with similar properties.

Our experiments also show that the proposed QNN-INT model can match or surpass the performance of classical CNN models on different datasets and with different parameter configurations. When compared with a standard quanvolutional model with rotational encoding (QNN-ROT), QNN-INT-SIMPLE and QNN-INT-RNDMUL surpassed its performance on all tested configurations.

The proposed integrated encoding model features a large number of hyperparameters, including the number of qubits n, the filter size k, and the number of gates L, in addition to an extensive range of possible mapping functions $\alpha(\cdot)$. Each parameter can be selected independently based on the hardware constraints of current quantum devices. The choice of the mapping function seems to be crucial to ensure higher expressivity and better classification results. However, this flexibility also introduces challenges: certain configurations, such as using RNDLIN as the mapping function, may negatively impact model performance. Future work could explore automated strategies for hyperparameter and mapping function selection, drawing inspiration from techniques developed for quantum architecture discovery [31], [32].

An exciting future direction would be to apply integrated encoding in a learnable setting. In particular, mapping functions such as the ones used in QNN-INT-RNDLIN and QNN-INT-RNDMUL have randomly initialized parameters, which could be optimized during training, as done in [19] for QNN-ROT. Such extensions could bridge the gap between current NISQ-compatible models and more expressive parameterized quantum circuits. While the current study focuses on nontrainable (random) quantum layers due to their alignment with NISQ-era constraints and classification within the $\mathrm{CSIM}_{\mathrm{QE}}$ class, this choice limits task-specific adaptability. On the other hand, parameterized quantum models offer improved learnability and flexibility but may be more resource-intensive and susceptible to issues such as barren plateaus. Studying this tradeoff between generalization and adaptability is a promising direction for future research.

APPENDIX A
EVALUATION OF PATCH DISTRIBUTION

To better understand whether local structural differences correlate with the performance trends observed in Section VI-C, we evaluate the distribution of image patches across different patch sizes $k \in \{2, 3, 4, 5\}$ for the two datasets: MiraBest and LArTPC.

For each dataset and value of k, we extract all unique $k \times k$ patches from the training set of each, flatten them and analyze the overall path distribution. We apply principal component analysis (PCA) to project the patch representations into three dimensions, allowing us to visualize the density and spread of patch types in a compact space.

Fig. 8 presents these distributions. Each point corresponds to a unique patch, its size indicates the frequency of the patch in the dataset, and its color indicates the ℓ_2 distance of the patch from the most frequent patch (the all-zero patch in both datasets) computed as $\|\mathbf{x} - \mathbf{0}\|_2$. This provides a visual cue for how distant each patch is from the "background."

For MiraBest, across all patch sizes, the distribution remains highly concentrated near the origin. This indicates limited structural diversity and high redundancy in local patterns. The patch space forms a compact cloud, suggesting that most regions in the images are visually uniform and carry low local complexity.

In LArTPC, as k increases, the patch distribution expands significantly. The PCA embedding forms a triangular or fan-shaped manifold, with patches populating more distant regions of the space and forming "clusters" of similar patches. This indicates that larger patches capture more diverse and informative local patterns, reflecting higher visual complexity in the data.

These findings offer a plausible explanation for the performance gap observed in Table 5: models trained on LArTPC benefit from the richer representational content present in its more diverse local patches, whereas those trained on MiraBest are limited by the homogeneity of local structures.

IEEE Transactions on Quantum Engineering

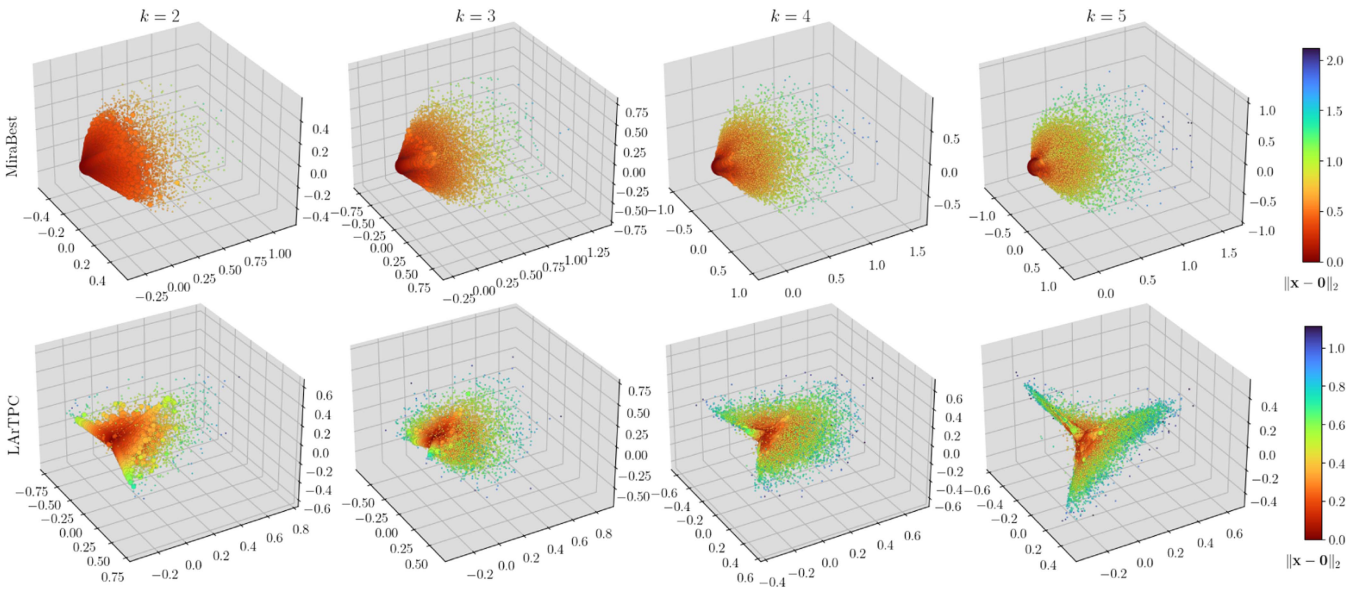

FIGURE 8. PCA projection of $k \times k$ patches extracted from MiraBest and LArTPC datasets. Each point corresponds to a unique patch; the size of the point indicates the frequency of the patch in the dataset; colors indicate distance from the most common (zero-valued) patch. As k increases, LArTPC exhibits a broader more diverse distribution, while MiraBest remains highly concentrated.

APPENDIX B
EMPIRICAL EXPRESSIBILITY ANALYSIS

The proposed QNN-INT encoding enables control over the expressibility of the ansatz by adjusting the number of gates L. To investigate the relationship between expressibility and model performance, we conducted experiments with a fixed kernel size of $k = 3$, varying L to assess whether changes in expressibility yield distinct behaviors. Specifically, we tested: Specifically, we tested:

1) $L = 9$ (low): the minimum required to encode all k^2 input features;
2) $L = 18$ (medium): the configuration used in the main experiments ($L = 2k^2$);
3) $L = 27$ (high): a more expressive configuration with additional gates.

Because increasing L also raises circuit complexity, we designed a complementary experiment using the QNN-ROT model. In this case, we varied the connection probability p to control circuit complexity while keeping expressibility roughly constant (see Fig. 7, bottom row). The tested values were:

1) $p = 0.075$ (low): reduced connectivity and circuit complexity;
2) $p = 0.15$ (medium): the baseline used in the main experiments;
3) $p = 0.225$ (high): higher connectivity and complexity.

We first examine the learning curves of all models (see Fig. 9) to understand how expressibility and circuit complexity influence training dynamics.

On the MiraBest dataset (top row), both QNN-INT-SIMPLE and QNN-INT-RNDMUL show slightly faster convergence and lower test loss with higher expressibility, albeit with a greater tendency to overfit. This suggests that increased expressibility improves representational capacity but may reduce generalization. In contrast, the QNN-ROT model shows minimal variation in training behavior across the different p values.

On the LArTPC dataset (bottom row), the same trend is observed but less pronounced. Higher expressibility leads to marginal improvements for QNN-INT-SIMPLE and QNN-INT-RNDMUL, while QNN-ROT shows benefits with increased circuit complexity, possibly due to more available parameters. Notably, results for QNN-INT-RNDLIN remain too noisy on both datasets to draw meaningful conclusions.

We also report test accuracy for each configuration in Table 6. On MiraBest, higher expressibility correlates weakly with improved performance in QNN-INT (e.g., QNN-INT-SIMPLE improves from 55.14 to 57.50). However, on LArTPC, models with Medium expressibility seem to perform better than the others, contradicting the initial hypothesis.

In addition, the QNN-ROT model shows a marked increase in performance moving from low to high circuit complexity (from 52.64 to 56.14 on MiraBest and from 71.55 to 75.32 on LArTPC), despite having a roughly constant expressibility.

In conclusion, while expressibility can influence model dynamics and performance, its role appears limited and context-dependent. Circuit complexity may play a more significant role than expressibility in determining performance for QNNs under the architectural and dataset constraints explored in this study.

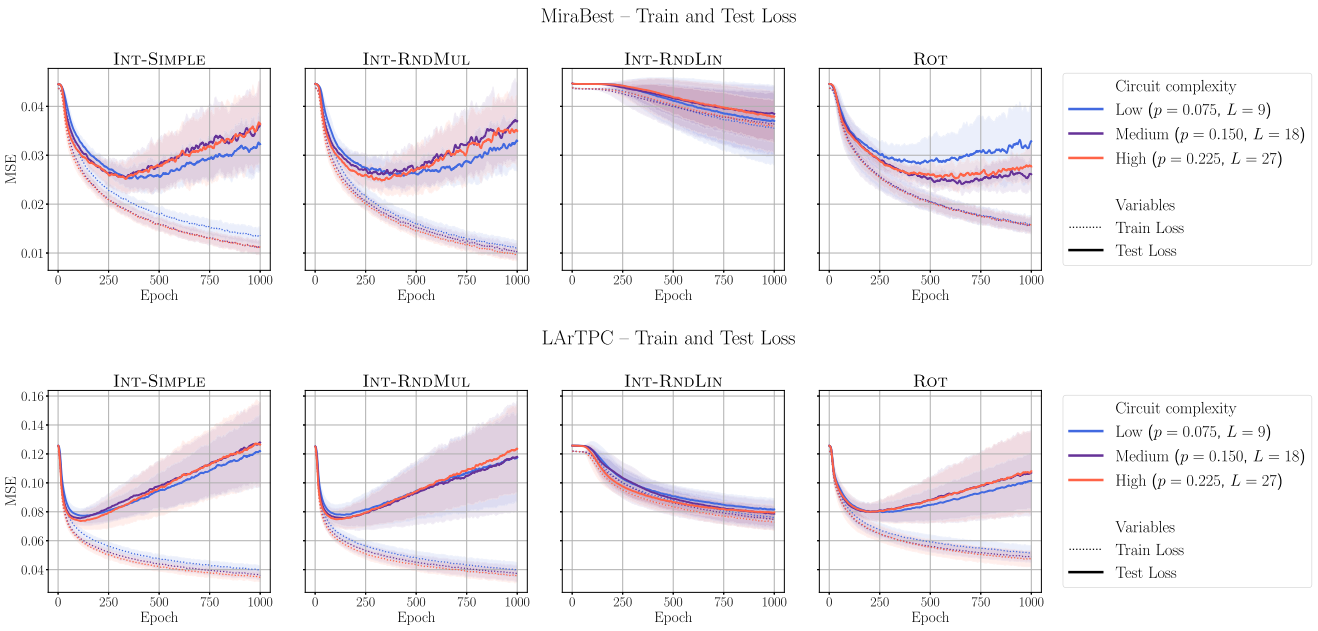

FIGURE 9. Average train and test losses (solid lines and dotted lines) for all models with different circuit complexities (different values of L for QNN-INT, and of p for QNN-ROT). The data reported are averages over ten runs, with the standard deviation represented by the shadowed area. Results are reported for the datasets MiraBest (top) and LArTPC (bottom).

TABLE 6. (Left) Average Final Test Accuracy for MiraBest and (Right) LArTPC Over Ten Seeds for Different Circuit Complexities

Model	Low $p = 0.075$ $L = 9$	Medium $p = 0.150$ $L = 18$	High $p = 0.225$ $L = 27$
QNN-INT-SIMPLE	55.14 ± 3.39	56.78 ± 2.80	57.50 ± 2.05
QNN-INT-RNDLIN	48.78 ± 15.13	46.57 ± 18.76	54.35 ± 4.74
QNN-INT-RNDMUL	56.14 ± 2.65	59.07 ± 3.70	57.28 ± 3.15
QNN-ROT	52.64 ± 3.97	53.78 ± 1.65	56.14 ± 3.12

(a) MiraBest

Model	Low $p = 0.075$ $L = 9$	Medium $p = 0.150$ $L = 18$	High $p = 0.225$ $L = 27$
QNN-INT-SIMPLE	77.40 ± 2.81	79.09 ± 2.55	77.79 ± 8.44
QNN-INT-RNDLIN	58.18 ± 7.39	58.05 ± 6.98	59.48 ± 8.04
QNN-INT-RNDMUL	78.70 ± 5.09	79.61 ± 2.67	79.87 ± 2.54
QNN-ROT	71.55 ± 6.46	73.89 ± 7.15	75.32 ± 1.73

(b) LArTPC

APPENDIX C
EFFECT OF NOISE ON INFERENCE

In this section, we perform a small-scale evaluation on simulated noisy hardware to assess the effect of noise on the proposed method. We use the FakeTorino noisy simulator, which mimics the noise characteristics of IBM's Torino hardware, a recent 133-qubit system based on the R1 Heron quantum processor. For each unique patch and each quantum circuit, we first transpile the logical circuit to the hardware coupling map and then simulate it while incorporating the device's noise model.

As noisy simulation is computationally expensive, we restrict our experiments to a single configuration. In particular, we select the best performing one from the main experiments: the QNN-INT-RNDMUL model with kernel size $k = 3$, on the MiraBest dataset. We test two training-evaluation setups: 1) training on a noiseless simulated device and evaluating on noisy images using the noisy simulator and 2) training and evaluating entirely in the noisy setting.

This mirrors practical deployment scenarios where models may be trained on high-performance low-noise systems but deployed for inference on hardware subject to realistic noise, as well as the more challenging case of noise-aware training.

TABLE 7. Comparison of Test Accuracy, Across Different Noise Settings on the MiraBest Dataset

Train Setting	Test Setting	Test Accuracy
Noiseless	Noiseless	80.52
Noiseless	Noisy	59.74
Noisy	Noisy	79.22

The results, reported in Table 7, show a clear impact of hardware noise on model performance. When both training and testing are performed on a noiseless simulator, the model achieves 80.52% accuracy, consistent with the results reported in the main experiments. However, when a model trained under noiseless conditions is evaluated on noisy hardware, accuracy drops to 59.74%, a decrease of more than 20 percentage points. In contrast, training the model directly on noisy simulations yields 79.22% accuracy when evaluated on noisy hardware, indicating that noise-aware training can substantially mitigate the detrimental effects of quantum hardware noise.

IEEE Transactions on Quantum Engineering

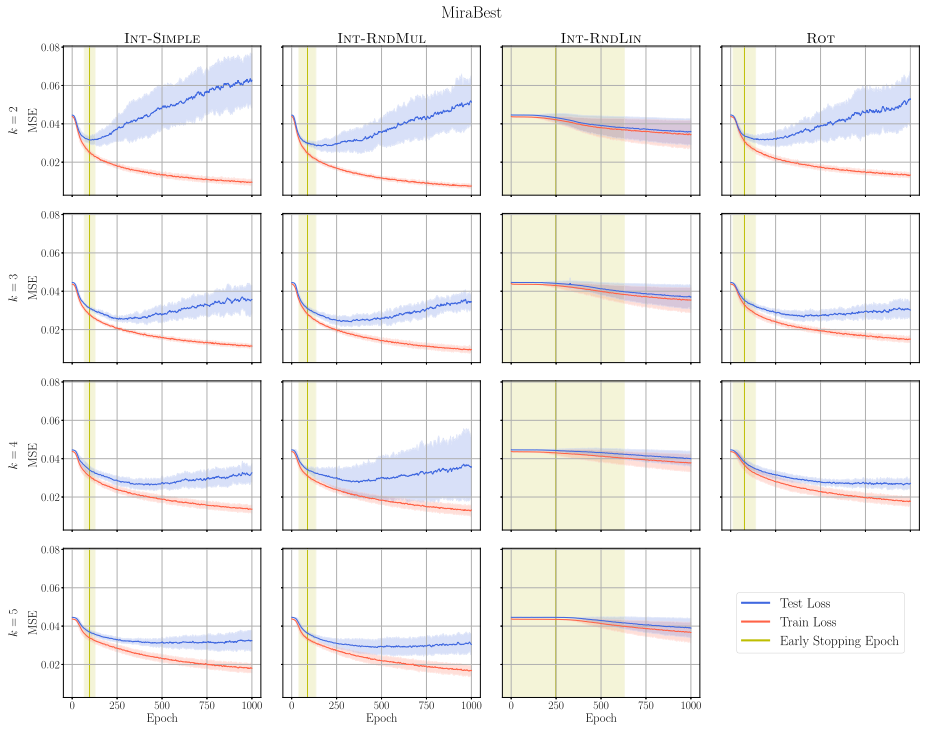

FIGURE 10. Average learning curves for the main experiments on MiraBest, consisting of train and test losses. Each row corresponds to a different kernel size *k*, and each column to a different model architecture. The vertical line represents the early stopping epoch. The shaded area indicates the standard deviation.

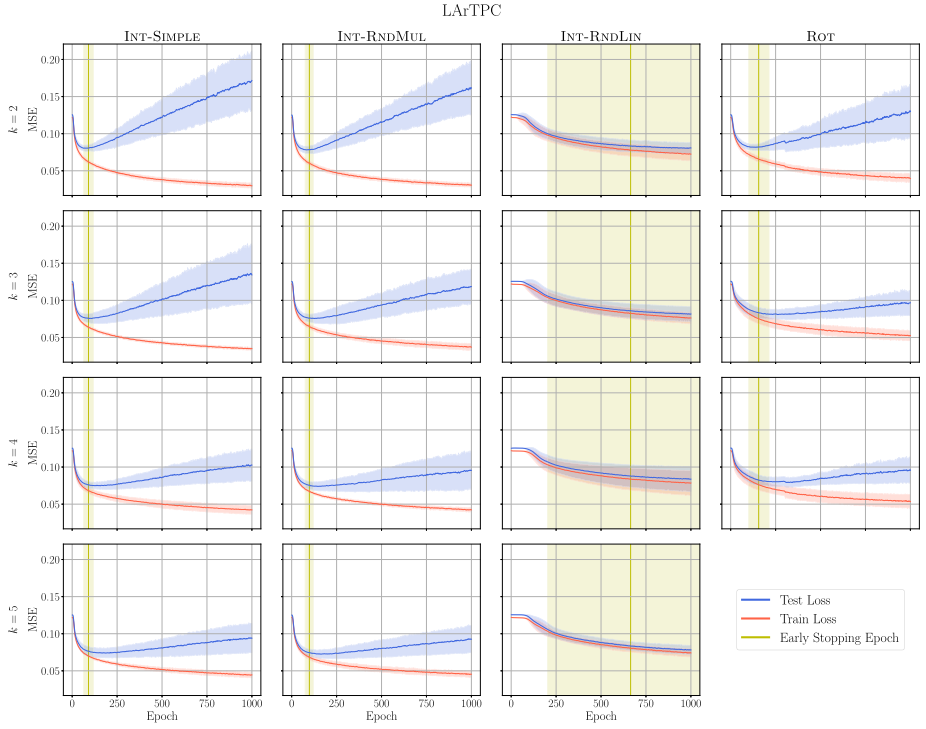

FIGURE 11. Average learning curves for the main experiments on LArTPC, consisting of train and test losses. Each row corresponds to a different kernel size *k*, and each column to a different model architecture. The vertical line represents the early stopping epoch. The shaded area indicates the standard deviation.

FIGURE 12. Learning curves for the noisy experiments on MiraBest, consisting of train and test losses. The vertical line represents the early stopping epoch.

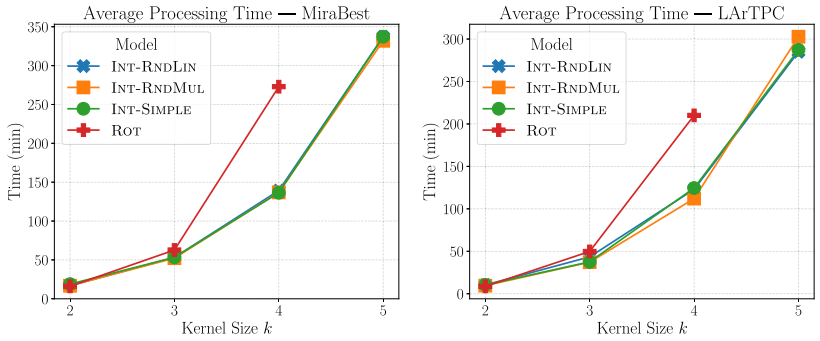

FIGURE 13. Average preprocessing time (in minutes) for each dataset, using eight layers and 50 levels of quantization with the noiseless simulation.

APPENDIX D
LEARNING CURVES AND SIMULATION TIME

For completeness, we show the learning curves of all the models used in the main experiments (see Figs. 10 and 11), as well as the additional noisy evaluation (see Fig. 12).

An interesting observation is that the models QNN-Int-Simple and QNN-Int-RndMul converge faster (see, for example, the test loss on LArTPC in Fig. 11). This suggests that these models have higher expressive power.

We also report the average time required to perform the quantum preprocessing of the datasets (see Fig. 13). The two datasets exhibit similar behavior, with differences mainly driven by the number of unique patches. The three QNN-Int models require very similar preprocessing times, as there are no structural differences in their circuits. The largest models require around 350 min to preprocess the MiraBest dataset and 300 min for the LArTPC dataset.

For kernel sizes of 2 and 3, almost no difference is observed between QNN-Rot and QNN-Int, despite the former requiring nine qubits and the latter four. This suggests that, when using the standard simulator implemented in qiskit, the integrated model can require more time than the rotational model when using the same number of qubits. However, for $k = 4$, the time for QNN-Rot increases steeply due to the cost of simulating circuits with 16 qubits. In contrast, the number of qubits in QNN-Int is fixed, leading to a quadratic increase in time with k, as the number of gates grows proportionally to k^2.

The noisy experiment reported in Appendix C required approximately 2.5 weeks to preprocess the MiraBest dataset (24 438 min).

ACKNOWLEDGMENT

Data availability: The code to generate and test quanvolutional models, with both rotational encoding and integrated model, can be obtained at https://github.com/Dan-LB/integrated_encoding_for_QuanvNN. The MiraBest dataset can be downloaded from [33]. The LArTPC dataset can be obtained by requesting it to the authors of [15].

REFERENCES

[1] R. Kharsa, A. Bouridane, and A. Amira, "Advances in quantum machine learning and deep learning for image classification: A survey," *Neurocomputing*, vol. 560, 2023, Art. no. 126843, doi: 10.1016/j.neucom.2023.126843.

[2] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," 2019, *arXiv:1905.13311*, doi: 10.48550/arXiv.1905.13311.

[3] J. Mcclean, S. Boixo, V. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nat. Commun.*, vol. 9, 2018, Art. no. 4812, doi: 10.1038/s41467-018-07090-4.

[4] M. P. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," *Quantum Mach. Intell.*, vol. 2, 2019, Art. no. 2, doi: 10.1007/s42484-020-00012-y.

[5] C.-H. H. Yang et al., "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6523–6527, doi: 10.1109/ICASSP39728.2021.9413453.

[6] S. Bhatta and J. Dang, "Multiclass seismic damage detection of buildings using quantum convolutional neural network," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 39, no. 3, pp. 406–423, Feb. 2024, doi: 10.1111/mice.13084.

[7] S. Sridevi, T. Kanimozhi, K. Issac, and M. Sudha, "Quanvolution neural network to recognize arrhythmia from 2D scaleogram features of ECG signals," in *Proc. Int. Conf. Innov. Trends Inf. Technol.*, 2022, pp. 1–5, doi: 10.1109/ICITIIT54346.2022.9744224.

[8] K. Zheng, J. Van Griensven, and R. Fraser, "A quantum machine learning approach to spatiotemporal emission modelling," *Atmosphere*, vol. 14, no. 6, 2023, Art. no. 944, doi: 10.3390/atmos14060944.

[9] M. Schuld and F. Petruccione, "Representing data on a quantum computer," in *Machine Learning With Quantum Computers*, Berlin, Germany: Springer, 2021, pp. 147–176, doi: 10.1007/978-3-030-83098-4_4.

[10] J. Zheng, Q. Gao, J. Lü, M. Ogorzałek, Y. Pan, and Y. Lü, "Design of a quantum convolutional neural network on quantum circuits," *J. Franklin Inst.*, vol. 360, no. 17, pp. 13761–13777, Nov. 2023, doi: 10.1016/j.jfranklin.2022.07.033.

[11] Y. Wang, Y. Wang, C. Chen, R. Jiang, and W. Huang, "Development of variational quantum deep neural networks for image recognition," *Neurocomputing*, vol. 501, pp. 566–582, Aug. 2022, doi: 10.1016/j.neucom.2022.06.010.

[12] A. Matic, M. Monnet, J. Lorenz, B. Schachtner, and T. Messerer, "Quantum-classical convolutional neural networks in radiological image classification," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, Sep. 2022, pp. 56–66, doi: 10.1109/QCE53715.2022.00024.

[13] A. Senokosov, A. Sedykh, A. Sagingalieva, B. Kyriacou, and A. Melnikov, "Quantum machine learning for image classification," *Mach. Learn.: Sci. Technol.*, vol. 5, no. 1, Mar. 2024, Art. no. 015040, doi: 10.1088/2632-2153/ad2aef.

[14] F. A. M. Porter and A. M. M. Scaife, "MiraBest: A data set of morphologically classified radio galaxies for machine learning," *RAS Techn. Instrum.*, vol. 2, no. 1, pp. 293–306, 2023, doi: 10.1093/rasti/rzad017.

[15] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, "Quantum convolutional neural networks for high energy physics data analysis," *Phys. Rev. Res.*, vol. 4, Mar. 2022, Art. no. 013231, doi: 10.1103/PhysRevResearch.4.013231.

[16] T. Hubregtsen, P. Josef, P. Stecher, and K. Bertels, "Evaluation of parameterized quantum circuits: On the relation between classification accuracy, expressibility, and entangling capability," *Quantum Mach. Intell.*, vol. 3, 2021, Art. no. 9, doi: 10.1007/s42484-021-00038-w.

[17] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, "Connecting ansatz expressibility to gradient magnitudes and barren plateaus," *PRX Quantum*, vol. 3, Jan. 2022, Art. no. 010313, doi: 10.1103/PRXQuantum.3.010313.

[18] A. Sebastianelli, D. A. Zaidenberg, D. Spiller, B. L. Saux, and S. Ullo, "On circuit-based hybrid quantum neural networks for remote sensing imagery classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 565–580, 2022, doi: 10.1109/JSTARS.2021.3134785.

[19] D. Mattern, D. Martyniuk, H. Willems, F. Bergmann, and A. Paschke, "Variational quanvolutional neural networks with enhanced image encoding," 2021, *arXiv:2106.07327*, doi: 10.48550/arXiv.2106.07327.

[20] M. Cerezo, et al., "Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing," *Nat. Commun.*, vol. 16, 2025, Art. no. 7907, doi: 10.1038/s41467-025-63099-6.

[21] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *Phys. Rev. Lett.*, vol. 122, 2018, Art. no. 040504, doi: 10.1103/PhysRevLett.122.040504.

[22] V. Havlíček et al., "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019, doi: 10.1038/s41586-019-0980-2.

[23] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, "Exponential concentration in quantum kernel methods," *Nat. Commun.*, vol. 15, no. 1, 2024, Art. no. 5200, doi: 10.1038/s41467-024-49287-w.

[24] P. Le, A. Iliyasu, F. Dong, and K. Hirota, "A flexible representation of quantum images for polynomial preparation, image compression and processing operations, quantum inf," *Quantum Inf. Process.*, vol. 10, pp. 63–84, 2011, doi: 10.1007/s11128-010-0177-y.

[25] B. Bollobás, *Random Graphs (ser. Cambridge Studies in Advanced Mathematics)*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2001, doi: 10.1017/CBO9780511814068.

[26] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Adv. Quantum Technol.*, vol. 2, no. 12, 2019, Art. no. 1900070, doi: 10.1002/qute.201900070.

[27] K. Zyczkowski and H.-J. Sommers, "Average fidelity between random quantum states," *Phys. Rev. A*, vol. 71, Mar. 2005, Art. no. 0 32313, doi: 10.1103/PhysRevA.71.032313.

[28] M. C. Caro et al., "Generalization in quantum machine learning from few training data," *Nat. Commun.*, vol. 13, no. 1, Aug. 2022, Art. no. 4919, doi: 10.1038/s41467-022-32550-3.

[29] Y. LeCun, C. Cortes, and C. Burges, *MNIST Handwritten Digit Database*. Atlanta, GA, USA: AT&T Labs, 2010. [Online]. Available: https://archive.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009, doi: 10.24432/C5889J.

[31] Y. Sun, Y. Ma, and V. Tresp, "Differentiable quantum architecture search for quantum reinforcement learning," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, Bellevue, WA, USA, 2023, pp. 15–19, doi: 10.1109/QCE57702.2023.10177.

[32] M. Incudini, D. Lizzio Bosco, F. Martini, M. Grossi, G. Serra, and A. Di Pierro, "Automatic and effective discovery of quantum kernels," *IEEE Trans. Emerg. Top. Comput. Intell.*, 23 Dec. 2024, doi: 10.1109/TETCI.2024.3499993.

[33] F. A. M. Porter, and M. M. Scaife, "MiraBest: a data set of morphologically classified radio galaxies for machine learning," *RAS Techn. Instrum.*, vol. 2121, pp. 293–306, 2023, doi: 10.1093/rasti/rzad017.

Daniele Lizzio Bosco was born in Caltagirone, Italy, in 1998. He received the B.S. degree in mathematics from the University of Udine, Udine, Italy, in 2021, and the double M.S. degree in artificial intelligence and cybersecurity from the University of Udine and the University of Klagenfurt, Klagenfurt, Austria, in 2023. He is currently working toward the Ph.D. degree in artificial intelligence with the University of Udine and the University of Naples Federico II, Naples, Italy, under the National Ph.D. Program.

He is a Member of the Artificial Intelligence Laboratory of Udine, University of Udine. He is part of the organization of the Annual European Summer School on Quantum AI. His main research interests include quantum machine learning, quantum optimization, and deep learning applied to environment and agriculture.

Beatrice Portelli is currently working toward the Ph.D. degree in artificial intelligence with the University of Udine, Udine, Italy, and the University of Naples Federico II, Naples, Italy, under the National Ph.D. Program.

She is a member of the Artificial Intelligence Laboratory of Udine, Udine. She works employing deep learning and natural language processing techniques. She has worked on several projects related to language models for adverse drug event extraction and normalization from social media texts, as well as fact verification models. Her current research interests include machine learning and deep learning methods for risk management in the agricultural–forestry sector.

Giuseppe Serra received the Ph.D. degree in computer engineering, multimedia and telecommunications from the University of Florence, Italy, in 2010.

He is an Associate Professor with the University of Udine, Udine, Italy, where he is leading the Artificial Intelligence Laboratory of Udine. His research interests include machine learning and deep learning.

Dr. Serra was a Lead Organizer of the International Workshop on Egocentric Perception, Interaction and Computing in 2016 and 2017. He gave tutorials at two international conferences: 2012 International Conference on Pattern Recognition and 2013 International Conference on Computer Analysis of Images and Patterns. He is an Editorial Board Member for IEEE TRANSACTIONS ON HUMAN MACHINE SYSTEMS and *ACM Transactions on Multimedia Computing, Communications, and Applications*. He was a Technical Program Committee Member of several conferences and workshops.