Quantum convolutional neural networks for high energy physics data analysis

Samuel Yen-Chi Chen,[1,*] Tzu-Chieh Wei,[2,†] Chao Zhang,[3,‡] Haiwang Yu,[3,§] and Shinjae Yoo[1,‖]

[1]*Computational Science Initiative, Brookhaven National Laboratory, Upton, New York 11973, USA*

[2]*C. N. Yang Institute for Theoretical Physics and Department of Physics and Astronomy, State University of New York at Stony Brook, Stony Brook, New York 11794-3840, USA*

[3]*Physics Department, Brookhaven National Laboratory, Upton, New York 11973, USA*

(Received 1 February 2021; accepted 29 January 2022; published 28 March 2022)

This paper presents a quantum convolutional neural network (QCNN) for the classification of high energy physics events. The proposed model is tested using a simulated dataset from the Deep Underground Neutrino Experiment. The proposed quantum architecture demonstrates an advantage of learning faster than the classical convolutional neural networks (CNNs) under a similar number of parameters. In addition to the faster convergence, the QCNN achieves a greater test accuracy compared to CNNs. Based on our results from numerical simulations, it is a promising direction to apply QCNN and other quantum machine learning models to high energy physics and other scientific fields.

DOI: 10.1103/PhysRevResearch.4.013231

I. INTRODUCTION

High energy physics (HEP) communities have a long history of working with large data and applying advanced statistics techniques to analyze experimental data in the energy, intensity, and cosmic frontiers. With ever-increasing data volumes, the HEP community needs a significant computational breakthrough to continue this trajectory, and tools developed in quantum information science (QIS) could provide a viable solution. Quantum supremacy is the potential to solve problems faster than any classical methods [1,2]. In computational-complexity-theoretic terms, this generally means providing a superpolynomial speedup over the best-known or possible classical algorithm [3]. In this paper, we shall use quantum advantage to denote that quantum devices have certain edges over classical ones, not necessarily with superpolynomial speedup.

Machine learning methods promise great benefits for scalable data analytics. The big wave of deep learning algorithm development stems from recent advances in convolutional neural networks (CNNs) [4–9], which can effectively capture spatial dependencies within an image, as well as automatically learn important features from them [6–8,10,11]. Along with big data and graphics processing unit (GPU) processing capabilities, deep learning has significantly improved the ability to analyze large volumes of images. There are several examples where CNNs have been successfully applied to HEP challenges using classical computers [12–17]. However, in quantum computing, no significant progress has been made toward implementing such robust representation learning methods to date. Despite this, it is important to explore the power of quantum machine learning, which has recently attracted substantial interests [18–22].

In this paper, we present a new hybrid quantum convolutional neural network (QCNN) framework to demonstrate the quantum advantage versus corresponding classical algorithms. We simulate its performance for classification of HEP events from the simulated data in neutrino experiments. We show that with a similar number of parameters in the QCNN and classical CNNs, the QCNN can learn faster or reach better testing accuracy with fewer training epochs. Thus, our simulations demonstrate potentially an empirical quantum advantage of QCNNs over CNNs in terms of testing accuracy. This paper is organized as follows: Section II introduces the HEP experimental data used in this paper. In Secs. III and IV, we describe the new QCNN architecture in detail. Section V shows the performance of the QCNN on the experimental data. Finally we discuss the results and potential future works in Sec. VI and offer concluding remarks in Sec. VII.

II. HIGH ENERGY PHYSICS DATA

In this paper, we use simulated data from the Deep Underground Neutrino Experiment (DUNE) [23] to develop and test our QCNN algorithms on high-energy experiments. Hosted in the United States, DUNE is the next-generation, international, world-class experiment to reveal new symmetries of nature. DUNE's primary goals include searching for CP violation in the lepton sector, determining the neutrino mass ordering, performing precision tests of the three-neutrino paradigm, detecting supernova neutrino bursts, and searching for nucleon

*ychen@bnl.gov
†tzu-chieh.wei@stonybrook.edu
‡czhang@bnl.gov
§hyu@bnl.gov
‖sjyoo@bnl.gov

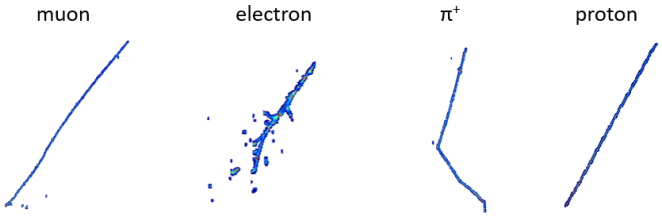

FIG. 1. Example images of simulated particle activities (μ^+, e^-, π^+, p) in a LArTPC detector. Colors in the images represent the sizes of the ionization energy loss along the particle trajectories when measured by LArTPC's wire planes.

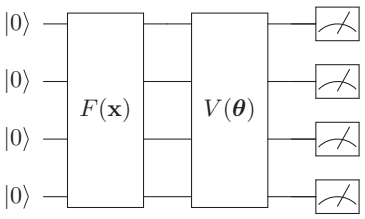

FIG. 2. General structure for the variational quantum circuit (VQC). The $F(\mathbf{x})$ is the quantum operation for encoding the classical data into the quantum state and $V(\boldsymbol{\theta})$ is the variational quantum circuit block with the adjustable parameters $\boldsymbol{\theta}$.

decays beyond the standard model. DUNE is an excellent test case for the QCNN because its main detector technology, the liquid argon time projection chamber (LArTPC), effectively provides high-resolution images of particle interactions as the ionized electrons drift toward the multiple sensing wire planes [24–27]. An advanced LArTPC simulation package, the Wire-Cell Toolkit [28] and LArSoft software [29], is used to generate realistic single-particle images in a LArTPC detector. The simulation implements a chain of algorithms, including: (1) generating single-particle kinematics, (2) applying LArTPC detector response, (3) adding realistic electronic noise, and (4) performing digital signal processing. Details about the LArTPC simulation can be found in Ref. [30]. Four different types of particles (μ^+, e^-, π^+, and p) are simulated. Figure 1 shows sample images of simulated particle activities on the collection wire plane. The images have a resolution of 480 × 600 pixels, where each pixel in the x axis represents a single wire and each pixel in the y axis represents a sampling time tick. In this paper, the goal of the QCNN is to predict the types of different particles by analogy with those performed via the classical CNN [16]. Each particle's momentum is set such that the mean range of the particle is about 2 meters, so the classification is not sensitive to the image size.

As visualized in Fig. 1, the classification of the four different particles primarily is a pattern recognition problem. A positively charged muon (μ^+) is a track-like particle, while an electron (e^-) produces electromagnetic showers that are spatially extended. A muon is a minimum ionizing particle in terms of energy loss along its trajectory, which translates into the intensity of the pixels. It experiences multiple Coulomb scattering (MCS) when passing the detector, causing its trajectory to deviate from a straight line. It also decays into a low-energy positron after it loses most of the kinetic energy and stops in the detector, leading to another short track segment near the end of the main track. A positively charged pion (π^+) looks similar to a muon in terms of energy loss, MCS, and decay, but it experiences additional nuclear interactions during its passage in the detector, often leading to a hard scattering (manifested as a "kink") along its main trajectory. Finally, a proton (p) also is a track-like particle. However, because a proton's mass is much heavier than a muon or pion, it has higher energy loss and encounters less MCS during travel. Consequently, a proton's track has higher intensity and is straighter than those from muons or pions.

These diverse features in detector images make the LArTPC data analysis well suited for CNN-type machine learning algorithms rather than hand-crafted feature extraction methods. Previous work with LArTPC has shown excellent performance from single-particle classification [16] to the more complicated neutrino interaction classification [31]. In this paper, we design and perform a quantum implementation of a classical CNN through variational quantum circuits (VQC) in LArTPC data analysis. We refer to this quantum CNN as QCNN. By comparing the performance to the classical CNN, we explore possible quantum acceleration and advantage in machine learning for HEP data analysis.

III. VARIATIONAL QUANTUM CIRCUITS

VQCs are quantum circuits that have *tunable* or *adjustable* parameters subject to classical iterative optimizations, which are commonly based on the gradient descent and its variants [22,32]. The general structure of a VQC is presented in Fig. 2. Here, the $F(\mathbf{x})$ block is for the state preparation that encodes the classical data $\mathbf{x}$ into the quantum state for the circuit to operate on and is not subject to optimization. This state preparation part is designed according to the given research problem. The $V(\boldsymbol{\theta})$ block represents the variational or *learning* part. The *learnable* parameters labeled with $\boldsymbol{\theta}$ will be optimized through gradient-based methods. For example, commonly used gradient-based optimizers are Adam [33] and RMSProp [34]. Conceptually, these parameters are comparable to the *weights* in classical deep neural networks (DNNs). In the final part of this VQC block, we perform the quantum measurement on a subset (or all) of the qubits to retrieve the information. If we run the circuit once and perform a single quantum measurement, it will yield a bit string, such as 0010, and it generally differs from what we will get if we prepare the circuit again and perform another quantum measurement due to the stochastic nature of quantum systems. However, if we prepare the same circuit and perform the quantum measurement multiple times, e.g., 1000 times, we will get the expectation values on each qubit, which should be quite close to the results from theoretical calculation. For example, consider a two-qubit system $|\Psi\rangle$, in every single measurement, the result is one of the following: 00, 01, 10, and 11. If we prepare $|\Psi\rangle$ and measure it 1000 times, we will get numerous measurements of 00, . . . , 11. We can count the frequencies of the appearances of 0 or 1 for each qubit and use them to estimate the expectation values. For example, if after 1000 repeated measurements, we get 600 measurements of 0 and 400 measurements of 1 in the first

qubit, then the expectation value of the first qubit is 0.4. In an N-qubit system, we place the expectation values of all qubits into a N-dimensional vector, which can be processed further in classical or quantum neural networks. We may choose different bases for the measurement. For example, in this paper, we exclusively use the Pauli-Z expectation values at the end of the VQC. Although VQCs are simple in concept, they are successful in machine learning tasks. Recent studies have reported the application of such variational architectures in the field of classification [18,19,21,22,35–40], function approximation [18,41,42], generative machine learning [43–47], metric learning [48,49], deep reinforcement learning [50–53], sequential learning [41,54], and speech recognition [55].

IV. QUANTUM CNN

CNNs have been tremendously successful in a wide spectrum of modern machine learning tasks, especially in the area of computer vision [4,6–8,10,11]. Such methods also allow new insights and progress in scientific research, for example, in HEP event classification [13–17] and phase transition studies [56]. With recent advances in quantum computing hardware [1,57,58], it is interesting to study the potential advantages and application scenarios for CNNs in the quantum regime. The QCNN is our proposed framework that uses VQCs to perform the convolutional operations. In this paper, we replace the classical neural-network-based convolutional filters, or *kernels*, with VQCs to harvest the expressive power granted by quantum entanglements. The quantum convolutional kernels will sweep through the input image pixels and transform them into a *representation vector* of lower dimensions by performing measurements (see Fig. 4).

A stack of VQCs will ensure features of varied length scales are captured in different layers.

A. Quantum Convolutional Filters

This section describes the VQC building blocks for the QCNN architecture.

1. Data Encoding Layer

In this layer, we first *encode* a classical input vector into a quantum state, which is necessary for additional processing. A general N-qubit quantum state can be represented as:

$$|\psi\rangle = \sum_{(q_1,q_2,\ldots,q_N)\in\{0,1\}^N} c_{q_1,\ldots,q_N}\,|q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \ldots \otimes |q_N\rangle\,, \tag{1}$$

where $c_{q_1,\ldots,q_N} \in \mathbb{C}$ is the *amplitude* of each quantum state and $q_i \in \{0, 1\}$. The square of the amplitude $c_{q_1,\ldots,q_N}$ is the *probability* of measurement with the post-measurement state in $|q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \ldots \otimes |q_N\rangle$, and the total probability should sum to 1, i.e.,

$$\sum_{(q_1,q_2,\ldots,q_N)\in\{0,1\}^N} \left|\left|c_{q_1,\ldots,q_N}\right|\right|^2 = 1. \tag{2}$$

In the proposed framework, the input to the VQC is a matrix with a dimension $n \times n$, where n is the filter or kernel size. The input will first be flattened and transformed into *rotation angles* for the quantum gates. In general, the input values of pixels are not in the interval of $[-1, 1]$. We use the arc tangent function to transform these input values into rotation angles. For each of the x_i in the $n \times n$ input, there will be two rotation angles generated, $\arctan(x_i)$ and $\arctan(x_i^2)$. This double encoding method is a standard method described in Ref. [18]. The $(n \times n)$-dimensional vector will then be transformed into $2n^2$ angles for the single-qubit rotation.

2. Variational Layer

After encoding the classical values into a quantum state, it will be subject to a series of unitary transformations. The variational layer (grouped in a dashed-line box in Fig. 5) consists of two parts. One is the *entanglement part*, which is a group of CNOT gates. The other is the *rotation part* that includes several single-qubit unitary rotations parameterized by 3 parameters α_i, β_i, and γ_i, where i represents the index of qubits. The parameters labeled by α_i, β_i, and γ_i are the ones that will be updated by the optimization procedure.

3. Quantum Measurement Layer

To obtain the transformed data from VQC blocks, we perform quantum measurements. We consider the ensemble samplings (expectation values) of the VQCs. If working with quantum simulation software (for example, PennyLane [59] or IBM Qiskit), this value can be calculated deterministically. While implementing on a real quantum computer, it is required to prepare the same system and carry out the measurements repeatedly to gain enough statistics. In the proposed QCNN architecture, the quantum convolutional filter will output a single value for each sweep step. Here, we perform the quantum measurements on the first qubit to get the expectation value.

B. Quantum Convolutional Operations

Both of the classical and quantum convolutional operations follow the following rule:

$$W_{\text{out}} = \frac{(W_{\text{in}} - F + 2P)}{S} + 1, \tag{3}$$

where

(i) W_{out} : the output dimension of the convolutional layer
(ii) W_{in} : the input dimension of the convolutional layer
(iii) F : the filter size
(iv) P : the padding size.

To capture the spatial dependency of the input data (e.g., images), the convolutional filter will sweep across the pixels and output the corresponding values at each location (see Fig. 3). In the QCNN, the filter itself is a VQC, which will transform an $n \times n$ dimensional vector into a single value (see Fig. 4). The circuit component for the quantum convolutional filter is in the Fig. 5. In general, each of the quantum convolutional filters captures a single kind of feature. We may place several filters in a convolutional layer to extract multiple features. In addition, we can *stack* multiple convolutional layers to extract different levels of features.

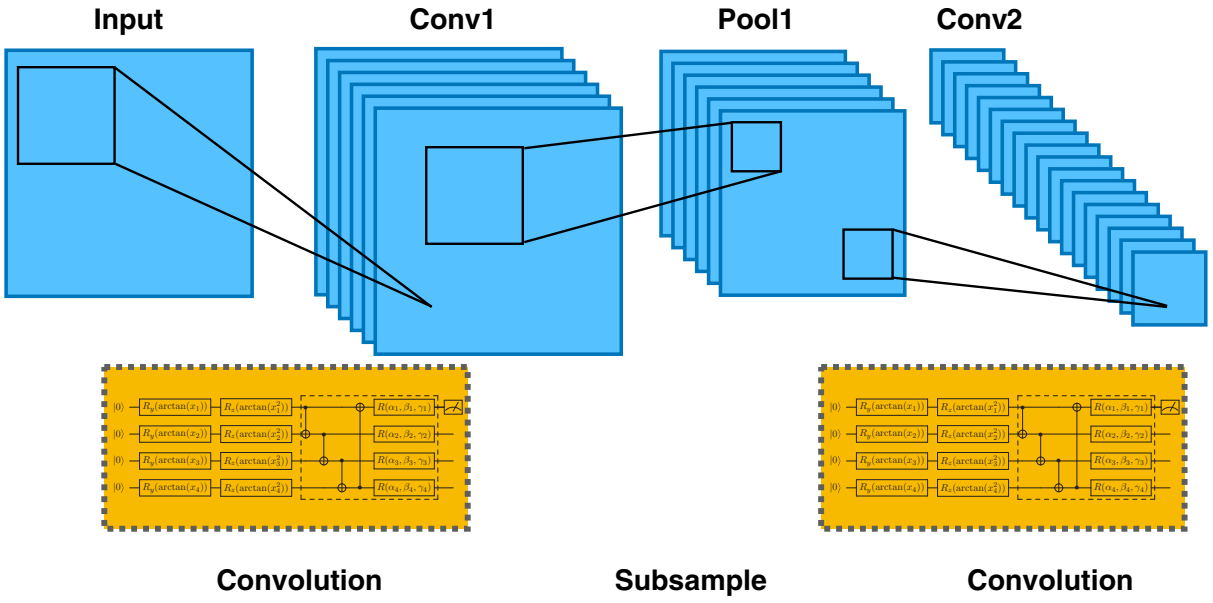

FIG. 3. Quantum CNN architecture. In the proposed hybrid quantum-classical model, the *filter* or *kernel* is a variational quantum circuit as shown in Fig. 5. Classical pooling and nonlinear activation functions can be optionally added between the convolutional layers analogous to the classical CNN.

C. Classical Post-processing

The output from the last quantum convolutional layer will then be flattened and processed by a single layer of a fully connected classical neural network. To represent the output values as the probabilities of each class label, we further employ the softmax function on the post-processed output.

D. Loss Function and Optimization

In this classification task, we use the *categorical cross-entropy* loss, which can be written in the following formulation:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{c=1}^{M} y_{o,c} \log(\hat{y}_{o,c}), \tag{4}$$

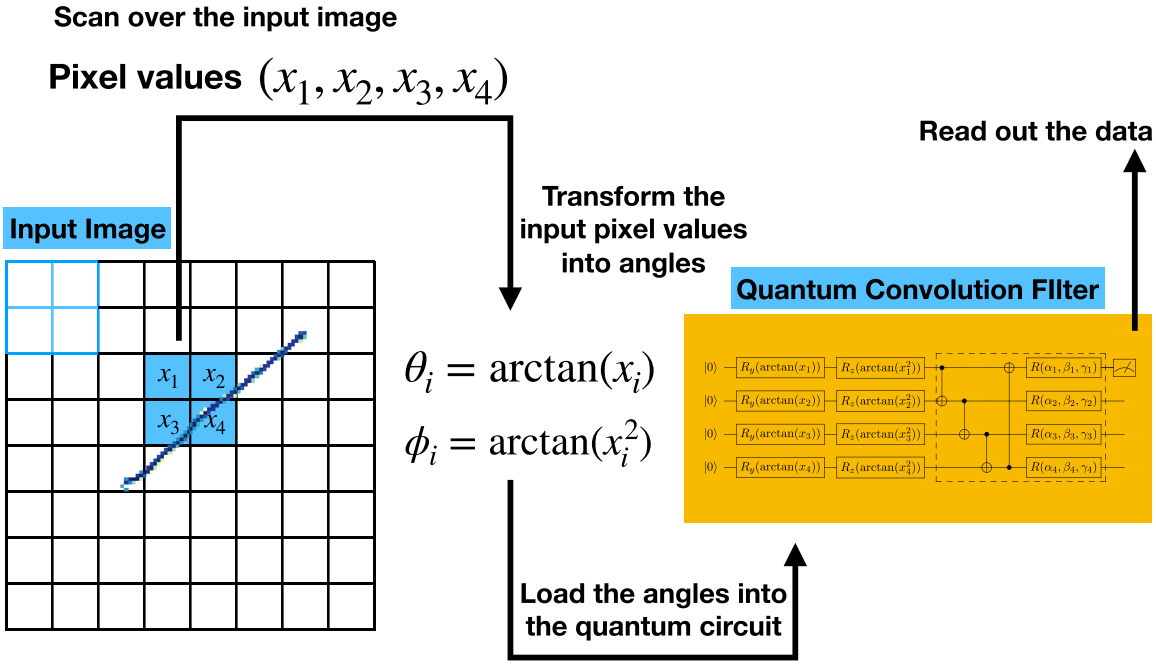

FIG. 4. QCNN operation. In the QCNN operation, the input pixel values (x_1, x_2, x_3, x_4) will first be encoded into a quantum state via the variational encoding method. Each value x_i is mapped into two values $\arctan(x_i)$ and $\arctan(x_i^2)$ for the R_y and R_z rotation angles, respectively. The quantum gates parameterized by α_i, β_i, γ_i then act on this encoded state. At the end of the circuit, Pauli-Z expectation values are retrieved. The retrieved values can then be processed with another layer of quantum convolutional layer or other classical operations (e.g., pooling, nonlinear activation functions, or dropout).

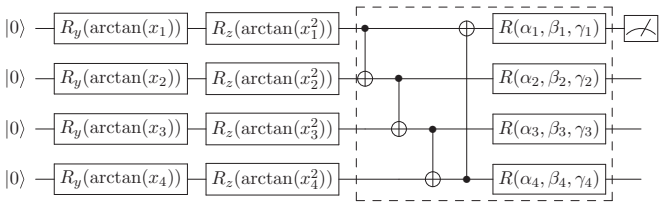

FIG. 5. Variational quantum circuit component for QCNN kernel (filter). The QCNN kernel (filter) includes three components: *encoding*, *variational*, and *quantum measurement*. The encoding component consists of several single-qubit gates $R_y(\arctan(x_i))$ and $R_z(\arctan(x_i^2))$, which represent rotations along y axis and z axis by the given angle $\arctan(x_i)$ and $\arctan(x_i^2)$, respectively. These rotation angles are derived from the input pixel values x_i and are not subject to iterative optimization. The choice of arc tangent function is motivated by the fact that in general the input values are not in the interval of $[-1, 1]$. The variational component consists of CNOT gates between each pair of neighboring qubits, which are used to entangle quantum states from each qubit and general single qubit unitary gates $R(\alpha, \beta, \gamma)$ with three parameters α, β, γ. Parameters labeled α_i, β_i, and γ_i are the ones for iterative optimization. The quantum measurement component will output the Pauli-Z expectation values of designated qubits. The number of qubits and the number of measurements can be adjusted to fit the problem of interest. In this paper, we use the VQC as a convolutional kernel (filter), therefore the number of qubits equals to the square of the kernel (filter) size and we only consider the measurement on the first qubit. The grouped box in the VQC may repeat several times to increase the number of parameters, subject to the capacity and capability of the available quantum computers or simulation software used for the experiments.

where

(i) M : the number of classes

(ii) log : the natural log

(iii) $y_{o,c}$: the binary indicator (0 or 1) if class label c is the correct classification for observation o

(iv) $\hat{y}_{o,c}$: predicted probability observation o is of class c.

In this paper, we use the gradient-based method to update the circuit parameters. The first problem is to calculate the gradients of quantum functions. The quantum functions are a series of operations with quantum gates, which are not the same as the layer operations in classical DNNs. In addition, the quantum functions typically are measured to retrieve the expectation values, which are stochastic by nature. In our paper, we adopt the *parameter-shift* rule [32,59] to perform all of the quantum gradient calculations. For example, if we know how to calculate the expectation value of an observable $\hat{P}$ on our quantum function,

$$\begin{aligned} f(x;\theta_i) &= \langle 0|U_0^\dagger(x)U_i^\dagger(\theta_i)\hat{P}U_i(\theta_i)U_0(x)|0\rangle \\ &= \langle x|U_i^\dagger(\theta_i)\hat{P}U_i(\theta_i)|x\rangle, \end{aligned} \tag{5}$$

where x is the input value (e.g., pixel values); $U_0(x)$ is the state preparation routine to transform or encode x into a quantum state; i is the circuit parameter index for which the gradient is to be evaluated; and $U_i(\theta_i)$ represents the single-qubit rotation generated by the Pauli operators X, Y, and Z. It can be shown [18] that the gradient of this quantum function f with respect to the parameter θ_i is

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}\left[f\left(x;\theta_i + \frac{\pi}{2}\right) - f\left(x;\theta_i - \frac{\pi}{2}\right)\right]. \tag{6}$$

Here, we have the recipe to calculate the quantum gradients. However, it still is not clear how to update the circuit parameters. In the simplest form of the gradient-descent method, the parameters are updated according to:

$$\theta \leftarrow \theta - \eta\nabla_\theta L(x;\theta), \tag{7}$$

where the θ is the model parameter, L is the loss function, and η is the learning rate. However, this vanilla form does not always work. For example, it may be easily stuck in local optimum [60], or it can make the model difficult to train. There are several gradient-descent variants that are successful [33,34,60]. Based on previous papers [41,50], we use the RMSProp optimizer to optimize our hybrid quantum-classical model. RMSProp [34] is a special kind of gradient-descent method with an adaptive learning rate that updates the parameters θ as:

$$E[g^2]_t = \alpha E[g^2]_{t-1} + (1-\alpha)g_t^2, \tag{8a}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}g_t, \tag{8b}$$

where g_t is the gradient at step t and $E[g^2]_t$ is the weighted moving average of the squared gradient with $E[g^2]_{t=0} = g_0^2$. The hyperparameters are set for all experiments in this paper as follows: learning rate $\eta = 0.01$, smoothing constant $\alpha = 0.99$, and $\epsilon = 10^{-8}$.

E. Dropout

Overfitting is a phenomenon where a machine learning model learned the statistical noise in the training data. This will cause poor performance when the models are tested against the unseen data or testing data. In other words, the model is not well generalizable. Such difficulties often emerge when training a classical DNN on a relatively small training set, and a QCNN is no exception.

One potential method to reduce overfitting is to train all possible neural network architectures on the given dataset and average the predictions from each model. However, this is impractical because it will require unlimited computational resources. *Dropout* is a method that approximates the effect of training a large number of neural networks with different architectures simultaneously [61].

The dropout operation entails (as follows): During the *training* phase, on each of the forward passes, some of the output values from the specified layer will become zeros. Each one of the values from the specified layer will be zeroed independently with probability p from a Bernoulli distribution. This dropout procedure will not be performed in the *testing* phase.

V. EXPERIMENTS AND RESULTS

This section presents the numerical simulation of QCNN on the task of classifying different HEP events. The input data have dimension 30×30. To demonstrate the possible quantum advantage, the classical and quantum CNN have a

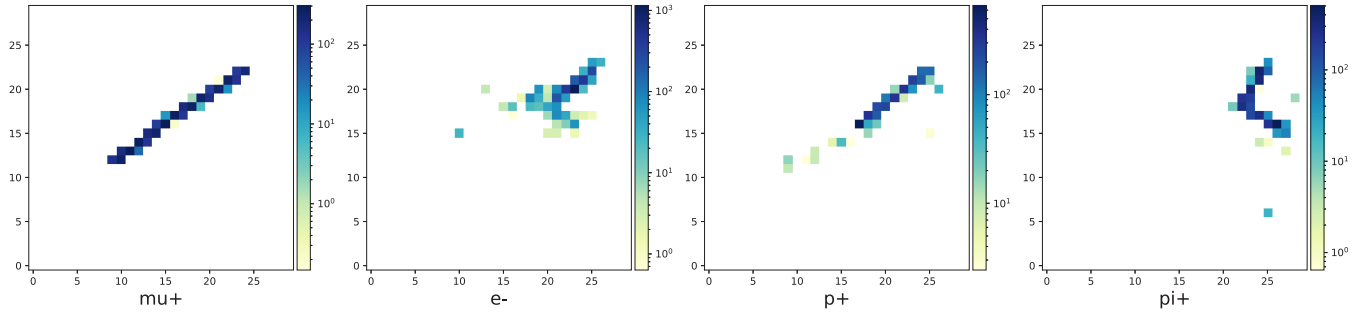

FIG. 6. Examples of scaled images of simulated particle activities (μ, π^+, p, e^-) in a LArTPC detector. These are the images used in the QCNN experiments. The dimension of these images is 30×30 pixels.

comparable number of parameters. Figure 6 shows the examples of the data used to train and test the QCNN models. For fair comparison, we arrange the experiment of QCNN and CNN to have similar numbers of parameters. In the classical CNN, there are 4 channels in the first convolutional layer with the filter size as 5×5 and 2 channels in the second convolutional layer with the filter size as 5×5. Finally, there is a fully connected layer, which features $7 \times 7 \times 2 \times 2 + 2 = 198$ parameters. Therefore, the total number of parameters in the classical CNN is $4 \times 5 \times 5 + 4 \times 2 \times 5 \times 5 + 198 = 498$. In the classical CNN, there is a dropout layer with dropout rate $= 0.5$ between the final convolutional layer and the fully connected layer. For the QCNN, there is 1 channel in the first quantum convolutional layer with the filter size of 3×3 and another single channel in the second quantum convolutional layer with the filter size of 2×2. Finally, there is a classical fully connected layer with $14 \times 14 \times 1 \times 2 + 2 = 394$ parameters. As such, the total number of parameters in the hybrid quantum-classical CNN is $54 + 24 + 394 = 472$ (see Table I). The software used for this paper are PyTorch [62], PennyLane [59], and Qulacs [63].

A. Muon versus Electron

Figure 7 and Table II depict the results of the classification between μ^+ and e^-. As mentioned in Sec. II, A μ^+ is a track-like particle, while an e^- produces electromagnetic showers that are spatially extended. This is a relatively straightforward pattern recognition problem when the particle tracks are more than a few meters ($\sim$2 meters in this simulation). In fact, we see that the test accuracy in QCNN (92.5%) and CNN (95%) is comparable to each other with a comparable number of parameters. On the other hand, the QCNN converges to its optimal accuracy much faster with a fewer number of epochs.

B. Muon versus Proton

Figure 8 and Table III show the results of the classification between μ^+ and p. As described in Sec. II, a proton is a track-like particle akin to a muon. However, because a proton's mass is much heavier than a muon or pion, it has higher energy loss and encounters less MCS when it passes the detector. As a result, a proton's track has higher intensity and is straighter than that of a muon. This classification is more difficult than

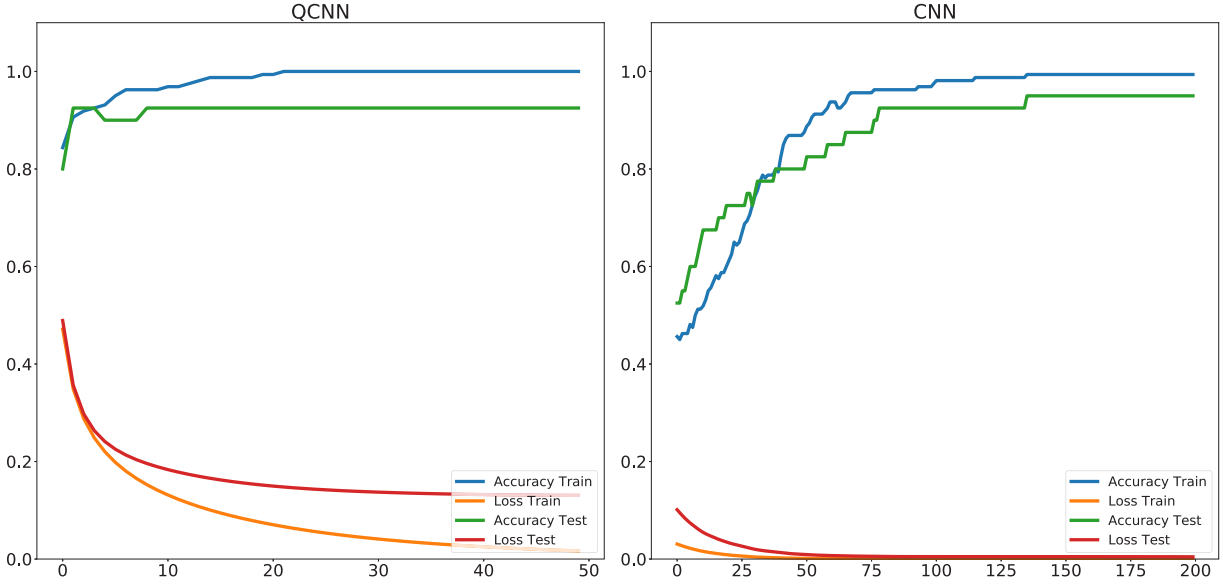

FIG. 7. QCNN on binary classification of muon vs electron. Training the QCNN for the classification of μ^+ and e^-. The filter size is 3 in the first convolutional layer and 2 in the second convolutional layer. There is 1 channel in both convolutional layers. The numbers of parameters in this setting are $9 \times 3 \times 2 = 54$ in the first convolutional layer, $4 \times 3 \times 2 = 24$ in the second convolutional layer, and $14 \times 14 \times 1 \times 2 + 2 = 394$ in the fully connected layer. The total number of parameters is $54 + 24 + 394 = 472$.

TABLE I. The comparison of model architectures between QCNN and CNN used in this paper.

	First conv layer (# of channels)	Filter size	Second conv layer (# of channels)	Filter size	Classical part (# of params)	Total number of params
QCNN	1	3×3	1	2×2	394	472
CNN	4	5×5	2	5×5	198	498

the previous case of muon versus electron, which is evident from the CNN's test accuracy of 80%. In this case, we show that with a comparable number of parameters, the QCNN outperforms the classical CNN, both in test accuracy and learning speed. The QCNN reaches 97.5% test accuracy at around 10 epochs, while the classical CNN plateaus at a test accuracy of 80% at roughly 75 epochs.

C. Muon versus Charged Pion

Figure 9 and Table IV illustrate the results of the classification between μ^+ and π^+. This is another difficult classification problem because a charged pion behaves much like a muon in terms of energy loss, MCS, and decay. As described in Sec. II, the main difference is that the π^+ experiences additional nuclear interactions during its passage through the detector, often leading to a hard scattering (manifested as a "kink") along its main trajectory. In this case, we show that with a comparable number of parameters, the QCNN outperforms the classical CNN, both in test accuracy and learning speed. The QCNN reaches 97.5% test accuracy at the first few epochs, while the classical CNN plateaus at a test accuracy of 82.5% at around 100 epochs. In this case, which is much more difficult than the previous two, we add a dropout layer in the QCNN with a dropout rate of 0.3 to improve its robustness against overfitting. We observe that the dropout operation significantly reduce the overfitting problem in this case. The testing accuracy is higher and the testing loss does not increase if we include dropout.

VI. DISCUSSION

A. Related Works

The concept of QCNNs has been discussed recently. In Ref. [64], the authors propose an architecture based on the multiscale entanglement renormalization Ansatz (MERA) tensor network to perform the classification of quantum states. Our approach differs from this paper as we focus on the classical input data. In Refs. [65,66], the authors propose a QCNN framework to deal with the classical data, which is like our method in the sense of targets. However, those works require the operation of quantum random access memory (QRAM), which is difficult to implement on physical devices in the near term. In [67,68], the authors consider a more realistic architecture that also is hybrid quantum-classical. While in a similar vein, our paper differs from that research because it implements input data with much larger dimensions. In Ref. [68], the data have dimension 3×3, while in Ref. [67], the data have dimension 10×10. Our architecture is capable of dealing with dimensions up to 30×30. In Refs. [55,69], quantum circuits are randomly sampled and not subject to iterative optimization. In our paper, the quantum and classical parts are trained in an end-to-end fashion. The trainability of QCNNs is studied in the recent paper [70], indicating that QCNN optimization is more viable than other quantum neural network architectures. A recent paper [71] points out that quantum neural networks can be mapped into a quantum kernel learning problem when supervised learning is discussed. The discussion is based on single layer QNN without multiple QNN layers separated by quantum measurements. Whether or not a QCNN architecture described in this paper can be transformed into a kernel model is an interesting topic and is left for future investigation.

B. Hyperparameter Optimization

Hyperparameter optimization is a technique to fine-tune the deep neural networks. Common techniques include grid search and random search. Although it is possible to use hyperparameter optimization to further fine-tune the CNN, it will not be included in this paper for the following reason: If CNN can be fine-tuned, it looks like that QCNN should also be fine-tuned to have a fair comparison. However, for the existing simulation software, it is extremely difficult to deploy QCNN experiments for hyperparameter optimization in any reasonable time frame since in the process of hyperparameter optimization, a significant volume of model validation or training is needed. In our case, even with the GPU simulation backend, it took nearly 4 week to finish a single training/testing case. It is impractical to perform hyperparameter optimization at the current stage. This research direction is to be pursued when the quantum simulation software is much faster in the future. Another point, which needs to be clarified here is that why certain constraints are imposed on classical CNNs. The reason is that it is nearly impossible at the moment to find a QML model (due to limited quantum devices and resources), which can beat a classical ML model

TABLE II. Performance comparison between the QCNN and the CNN on the binary classification between μ^+ and e^-.

	Training accuracy	Testing accuracy	Training loss	Testing loss
QCNN	100%	92.5%	0.017	0.13
CNN	99.38%	95%	0.0002	0.0046

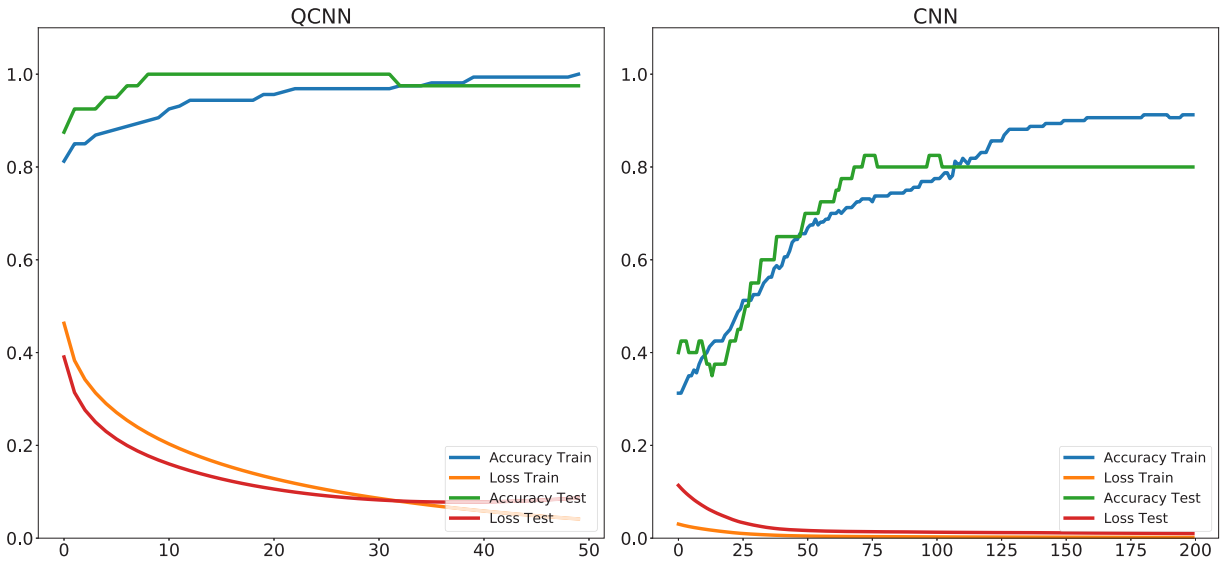

FIG. 8. QCNN on binary classification of muon vs proton. Training the QCNN for the classification of μ^+ and p. The filter size is 3 in the first convolutional layer and 2 in the second convolutional layer. There is 1 channel in both convolutional layers. The numbers of parameters in this setting are $9 \times 3 \times 2 = 54$ in the first convolutional layer, $4 \times 3 \times 2 = 24$ in the second convolutional layer, and $14 \times 14 \times 1 \times 2 + 2 = 394$ in the fully connected layer. The total number of parameters is $54 + 24 + 394 = 472$.

TABLE III. Performance comparison between the QCNN and the CNN on the binary classification between μ^+ and p.

	Training accuracy	Testing accuracy	Training loss	Testing loss
QCNN	100.00%	97.5%	0.041	0.087
CNN	91.25%	80%	0.002	0.01

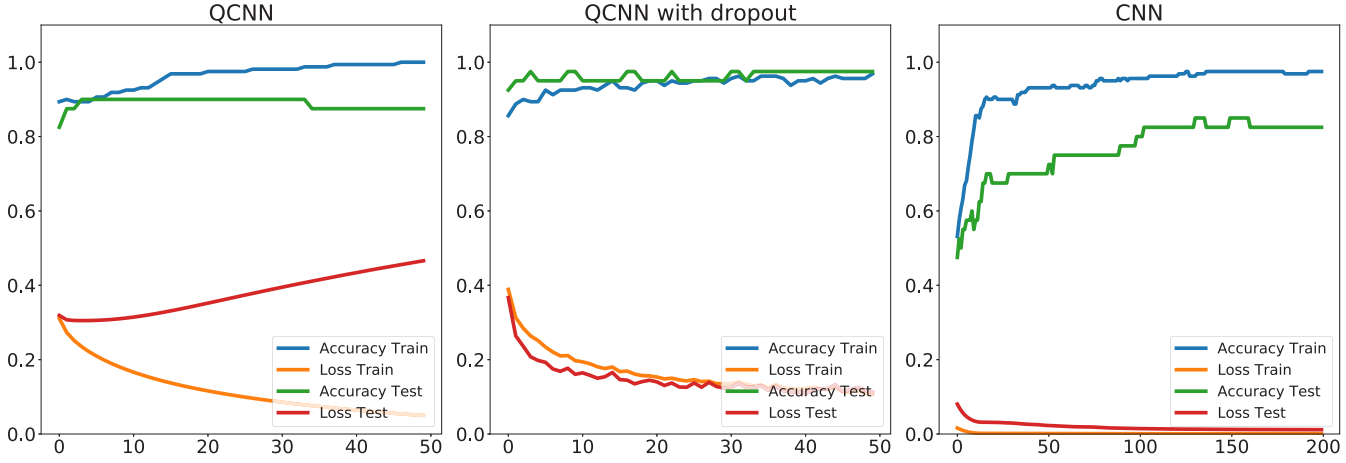

FIG. 9. QCNN on binary classification of muon versus charged pion. Training the QCNN for the classification of μ^+ and π^+. The filter size is 3 in the first convolutional layer and 2 in the second convolutional layer. There is 1 channel in both convolutional layers. The number of parameters in this setting are $9 \times 3 \times 2 = 54$ in the first convolutional layer, $4 \times 3 \times 2 = 24$ in the second convolutional layer, and $14 \times 14 \times 1 \times 2 + 2 = 394$ in the fully connected layer. The total number of parameters is $54 + 24 + 394 = 472$. In this experiment, we add a dropout layer with a dropout rate of 0.3 to the QCNN. We observe that the dropout operation significantly reduces the overfitting problem in this case.

TABLE IV. Performance comparison between the QCNN and the CNN on the binary classification between μ^+ an π^+.

	Training accuracy	Testing accuracy	Training loss	Testing loss
QCNN	100%	87.5%	0.05	0.47
QCNN (with dropout)	96.88%	97.5%	0.1066	0.1121
CNN	97.5%	82.5%	0.0006	0.0116

if there are no certain constraints. It is nearly trivial to build a classical CNN with millions of parameters on a personal computer and beat all existing quantum models. However, as the number of qubits increases and the gate fidelity improves, quantum machine learning is expected to provide substantial advantages and thus our paper shows the value of potential implication of the quantum advantages.

VII. CONCLUSION AND OUTLOOK

In this paper, we propose a quantum machine learning framework for learning HEP events. The particular dataset used in this paper is to explore whether there is any empirical quantum advantage provided by the QCNN. The dataset itself is classical and in the real experiments they are collected by sensors or cameras. The reason to study in this direction is that the data resolution may grow larger and larger, and in the future, it is possible that a large-scale quantum computer may be available and can be used to demonstrate quantum advantages on HEP data analysis. Specifically, we demonstrate that the QCNN architecture has a significant learning capacity in terms of learning speed and testing accuracy compared to the classical CNN when both use a comparable number of parameters. We expect the proposed framework will have a wide range of applications in the era of noisy intermediate-scale quantum (NISQ) devices and beyond, as well as in more HEP experiments.

Looking ahead, there are several research areas where we could extend our QCNN framework. First, in this paper, we perform experiments using the input dimension of 30×30 and a single input channel. In comparison, multiple input channels (e.g., different color channels) with higher dimensions are rather common in classical CNNs. Future studies to increase the data complexity in QCNNs are expected as the speed of quantum simulators improves. Second, this paper presents a noise-free simulation to demonstrate proof-of-concept QCNN experiments on HEP event classification. Our framework is based on VQC, which has the potential to be robust against device noise. However, applying parameter-shift methods to calculate quantum gradients requires a large amount of quantum circuit evaluations, which is infeasible at this time. For example, given a filter with the size $N \times N$ and the number of operations needed to scan over a single input images S, the contribution to the total number of circuit evaluation is, at least, $\mathcal{O}(N^2 S)$. Therefore, the number of total circuit evaluations grows as the circuit goes deeper (with more convolutional layers) or expands wider (with more filters in each layer). We reserve pursuing this area of study until the appropriate quantum computing resources are available. Finally, as the convolutional operation is quite versatile, it has been widely used beyond computer vision in classical machine learning, for example, in modeling data with temporal or sequential dependencies [72–76]. As such, our proposed general QCNN architecture is not limited to image classification tasks and can be extended to other application domains.

ACKNOWLEDGMENTS

This work is supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics program under Award No. DE-SC-0012704 and the Brookhaven National Laboratory Directed Research and Development (LDRD) Program No. 20-024.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

[2] A. W. Harrow and A. Montanaro, Quantum computational supremacy, Nature (London) **549**, 203 (2017).

[3] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2002).

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE **86**, 2278 (1998).

[5] D. Ciregan, U. Meier, and J. Schmidhuber, Multi-column deep neural networks for image classification, in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2012), pp. 3642–3649.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (NIPS, 2012), pp. 1097–1105.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2015), pp. 1–9.

[8] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

[9] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2016), pp. 770–778.

[10] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[11] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning* (MIT Press, Cambridge, 2016), Vol. 1.

[12] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, Machine learning at the energy and intensity frontiers of particle physics, Nature (London) **560**, 41 (2018).

[13] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, Jet-images-deep learning edition, J. High Energy Phys. 07 (2016) 069.

[14] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, Deep learning in color: towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.

[15] J. S. H. Lee, I. Park, I. J. Watson, and S. Yang, Quark-gluon jet discrimination using convolutional neural networks, J. Korean Phys. Soc. **74**, 219 (2019).

[16] R. Acciarri *et al.*, Convolutional neural networks applied to neutrino events in a liquid argon time projection chamber, J. Instrum. **12**, P03011 (2017).

[17] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, and P. Vahle, A convolutional neural network neutrino event classifier, J. Instrum. **11**, P09001 (2016).

[18] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[19] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, Circuit-centric quantum classifiers, Phys. Rev. A **101**, 032308 (2020).

[20] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[21] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002.

[22] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. **4**, 043001 (2019).

[23] B. Abi *et al.*, Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume II DUNE Physics, 2, 2020.

[24] C. Rubbia, The liquid argon time projection chamber: A new concept for neutrino detectors, CERN-EP-INT-77-08, CERN-EP-77-08, 5, 1977.

[25] H. H. Chen, P. E. Condon, B. C. Barish, and F. J. Sciulli, A Neutrino detector sensitive to rare processes. I. A Study of neutrino electron reactions, FERMILAB-PROPOSAL-0496, 1976.

[26] W. Willis and V. Radeka, Liquid argon ionization chambers as total absorption detectors, Nucl. Instrum. Meth. **120**, 221 (1974).

[27] D. Nygren, The time projection chamber: A new 4 pi detector for charged particles, eConf **C740805**, 58 (1974).

[28] B. Viren *et al.*, Wire-Cell Toolkit. https://github.com/WireCell/wire-cell-toolkit, 2020.

[29] E. L. Snider and G. Petrillo, LArSoft: Toolkit for simulation, reconstruction and analysis of liquid argon TPC neutrino detectors, J. Phys. Conf. Ser. **898**, 042057 (2017).

[30] C. Adams *et al.*, Ionization electron signal processing in single phase LArTPCs. Part I. Algorithm Description and quantitative evaluation with MicroBooNE simulation, J. Instrum. **13**, P07006 (2018).

[31] B. Abi *et al.*, Neutrino interaction classification with a convolutional neural network in the DUNE far detector, Phys. Rev. D **102**, 092003 (2020).

[32] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[33] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[34] T. Tieleman and G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4, 26–31 (2012).

[35] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, Transfer learning in hybrid classical-quantum neural networks, Quantum **4**, 340 (2020).

[36] Z. Abohashima, M. Elhosen, E. H. Houssein, and W. M. Mohamed, Classification with quantum machine learning: A survey, arXiv:2006.12270.

[37] P. Easom-McCaldin, A. Bouridane, A. Belatreche, and R. Jiang, Towards building a facial identification system using quantum machine learning techniques, arXiv:2008.12616.

[38] A. Sarma, R. Chatterjee, K. Gili, and T. Yu, Quantum unsupervised and supervised learning on superconducting processors, Quantum Inf. Comput. **20**, 541 (2020).

[39] S. A. Stein, B. Baheri, R. M. Tischio, Y. Chen, Y. Mao, Q. Guan, A. Li, and B. Fang, A hybrid system for learning classical data in quantum states, arXiv:2012.00256.

[40] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, Hybrid quantum-classical classifier based on tensor network and variational quantum circuit, in *Proceeding for the First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)* (NIPS, Vancouver, Canada, 2020).

[41] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, Quantum long short-term memory, arXiv:2009.01783.

[42] O. Kyriienko, A. E. Paine, and V. E. Elfving, Solving nonlinear differential equations with differentiable quantum circuits, Phys. Rev. A **103**, 052416 (2021).

[43] P.-L. Dallaire-Demers and N. Killoran, Quantum generative adversarial networks, Phys. Rev. A **98**, 012324 (2018).

[44] S. A. Stein, B. Baheri, R. M. Tischio, Y. Mao, Q. Guan, A. Li, B. Fang, and S. Xu, Qugan: A generative adversarial network through quantum states, arXiv:2010.09036.

[45] C. Zoufal, A. Lucchi, and S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, npj Quant. Info. **5**, 103 (2019).

[46] H. Situ, Z. He, L. Li, and S. Zheng, Quantum generative adversarial network for generating discrete data, Inf. Sci. **538**, 193 (2020).

[47] K. Nakaji and N. Yamamoto, Quantum semi-supervised generative adversarial network for enhanced data classification, Sci. Rep. **11**, 19649 (2020).

[48] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, arXiv:2001.03622.

[49] N. A. Nghiem, S. Y.-C. Chen, and T.-C. Wei, A unified classification framework with quantum metric learning, Phys. Rev. Res. **3**, 033056 (2021).

[50] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, Variational quantum circuits for deep reinforcement learning, IEEE Access **8**, 141007 (2020).

[51] O. Lockwood and M. Si, Reinforcement learning with quantum variational circuit, in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (AAAI, Palo Alto, CA, 2020), Vol. 16, pp. 245–251.

[52] S. Jerbi, L. M. Trenkwalder, H. P. Nautrup, H. J. Briegel, and V. Dunjko, Quantum Enhancements for Deep Reinforcement Learning in Large Spaces, PRX Quantum **2**, 010328 (2021).

[53] C.-C. Chen, K. Shiba, M. Sogabe, K. Sakamoto, and T. Sogabe, Hybrid quantum-classical Ulam-von Neumann linear solver-based quantum dynamic programing algorithm, in *Proceedings of the Annual Conference of JSAI* (Jpn. Soc. Artif. Intell., 2020), p. 2K6ES203.

[54] J. Bausch, Recurrent quantum neural networks, in *Advances in Neural Information Processing Systems 33* (NIPS, 2020), pp. 1368–1379.

[55] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition, in *ICASSP 2021-2021 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP) (IEEE, Piscataway, NJ, 2021), pp. 6523–6527.

[56] A. Tanaka and A. Tomiya, Detection of phase transition via convolutional neural networks, J. Phys. Soc. Jpn. **86**, 063001 (2017).

[57] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[58] A. Cross, The ibm q experience and qiskit open-source quantum computing software, in APS Meeting Abstracts, 2018.

[59] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv:1811.04968.

[60] S. Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Research **15**, 1929 (2014).

[62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems* (NIPS, 2019), pp. 8026–8037.

[63] Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya *et al.*, Qulacs: A fast and versatile quantum circuit simulator for research purpose, Quantum **5**, 559 (2021).

[64] I. Cong, S. Choi, and M. D. Lukin, Quantum convolutional neural networks, Nat. Phys. **15**, 1273 (2019).

[65] I. Kerenidis, J. Landman, and A. Prakash, Quantum algorithms for deep convolutional neural networks, arXiv:1911.01117.

[66] Y. Li, R.-G. Zhou, R. Xu, J. Luo, and W. Hu, A quantum deep convolutional neural network for image recognition, Quantum Sci. Technol. **5**, 044003 (2020).

[67] S. Oh, J. Choi, and J. Kim, A tutorial on quantum convolutional neural networks (QCNN), arXiv:2009.09423.

[68] J. Liu, K. H. Lim, K. L. Wood, W. Huang, C. Guo, and H.-L. Huang, Hybrid quantum-classical convolutional neural networks, China-Phys. Mech. Astron. **64**, 290311 (2021).

[69] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, Quanvolutional neural networks: powering image recognition with quantum circuits, Quantum Mach. Intell. **2**, 2 (2020).

[70] A. Pesah, M. Cerezo, S. Wang, T. Volkoff, A. T. Sornborger, and P. J. Coles, Absence of Barren Plateaus in Quantum Convolutional Neural Networks, Phys. Rev. X **11**, 041011 (2021).

[71] M. Schuld, Quantum machine learning models are kernel methods, arXiv:2101.11020.

[72] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, A convolutional neural network for modelling sentences, arXiv:1404.2188.

[73] B. Hu, Z. Lu, H. Li, and Q. Chen, Convolutional neural network architectures for matching natural language sentences, in *Advances in Neural Information Processing Systems* (NIPS, 2014), pp. 2042–2050.

[74] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, M. Adam, A. Gertych, and R. San Tan, A deep convolutional neural network model to classify heartbeats, Comput. Biol. Med. **89**, 389 (2017).

[75] S. Lai, L. Xu, K. Liu, and J. Zhao, Recurrent convolutional neural networks for text classification, in *Twenty-ninth AAAI Conference on Artificial Intelligence* (AAAI, Palo Alto, CA, 2015).

[76] W. Yin, H. Schütze, B. Xiang, and B. Zhou, ABCNN: Attention-based convolutional neural network for modeling sentence pairs, Trans. Assoc. Comput. Linguistics **4**, 259 (2016).