Classical versus quantum: Comparing tensor-network-based quantum circuits on Large Hadron Collider data

Jack Y. Araz[*] and Michael Spannowsky[†]
Institute for Particle Physics Phenomenology, Durham University, South Road, Durham DH1 3LE, England, United Kingdom

(Received 12 April 2022; accepted 1 November 2022; published 19 December 2022)

Tensor networks (TN) are approximations of high-dimensional tensors designed to represent locally entangled quantum many-body systems efficiently. This paper provides a comprehensive comparison between classical TNs and TN-inspired quantum circuits in the context of machine learning on highly complex, simulated Large Hadron Collider data. We show that classical TNs require exponentially large bond dimensions and higher Hilbert-space mapping to perform comparably to their quantum counterparts. While such an expansion in the dimensionality allows better performance, we observe that, with increased dimensionality, classical TNs lead to a highly flat loss landscape, rendering the usage of gradient-based optimization methods highly challenging. Furthermore, by employing quantitative metrics, such as the Fisher information and effective dimensions, we show that classical TNs require a more extensive training sample to represent the data as efficiently as TN-inspired quantum circuits. We also engage with the idea of hybrid classical-quantum TNs and show possible architectures to employ a larger phase space from the data. We offer our results using three main TN *Ansätze*: Tree tensor networks, matrix product states, and multiscale entanglement renormalization *Ansätze*.

DOI: 10.1103/PhysRevA.106.062423

I. INTRODUCTION

Machine learning (ML) and deep neural networks have gained tremendous interest on the verge of technological developments and the availability of an abundance of data. The high-energy physics (HEP) community has long been engaged with processing vast amounts of data generated by collider experiments like the Large Hadron Collider (LHC). Recent advancements in neural network (NN) technology allowed physics-driven analytic analysis methods to evolve into data-driven statistical approaches, which transformed the ability and accuracy of data analysis. The emergence of quantum computers has introduced another step of evolution in this formidable avenue.

Various quantum algorithms aim to tackle challenging tasks in optimization problems and improve the interpretability of classical NNs applied to high-energy physics data. Such algorithms include but are not limited to simulations of collision events [1–6], reconstruction of the charged tracks [7–10], and event classification analyses [11–19]. Despite the exceptional interest in quantum computation methods, there are still many open questions regarding the advantages of the quantum age [20].

There is particular interest in using the quantum paradigm in ML and optimization tasks, as the number of qubits needed is often already available in noisy intermediate-scale quantum devices, and error mitigation might be less of a concern. Quantum NNs have various advantages over classical NNs [14,16,21–24], such as faster convergence and significantly better performance with the same network structure. Quantum algorithms achieve this by representing the correlations between input features within the quantum entanglement paradigm, providing a much richer data representation.

Tensor networks (TNs) are widely used to simulate strongly entangled quantum systems [25,26], and they can represent both quantum states and quantum circuits (QCs) [27–29]. Due to this property, TNs form a natural bridge between classical and quantum computation methods. In the context of high-energy physics data analysis, tensor networks have recently been used for data originating from collider experiments. TN properties can be used to classify b-jets produced at the LHCb experiment where tabular data are analyzed using tree tensor network (TTN) architecture [30]. Moreover, Ref. [19] shows that matrix product states (MPSs) can achieve state-of-the-art convolutional neural network (CNN) accuracies in top tagging via calorimeter images. By employing entanglement entropy information between MPSs' tensor blocks the same accuracy can be achieved with only 54% of the pixels in a given calorimeter image.

Given the ability to represent both NNs and quantum many-body systems, it is only natural to use TN-inspired quantum circuit architecture to transfer our classical knowledge on quantum hardware [31–37]. Besides the high accuracy rates, TN-based quantum circuits can lead to more robust results against noise in the near-term quantum computers. Whilst this opens up an entirely new circuit design, near-term quantum devices are still limited to a small number of qubits, restricting the usage of multimodal datasets. Hybrid classical TNs (CTNs) and quantum TNs (QTNs) can be utilized to eliminate this issue to design end-to-end training sequences with classical data mapping and quantum classification layers [38,39]. This allows much larger datasets to be

[*]Corresponding author: jack.araz@durham.ac.uk
[†]michael.spannowsky@durham.ac.uk

embedded in the optimization process, and due to the duality of TNs, as more qubits are available, classical nodes can be transformed into quantum nodes to harness the full potential of the quantum hardware.

This paper investigates the usage of TN-based variational quantum circuits (VQCs) for top jet discrimination from the QCD background in calorimeter images. We have investigated three different architectures, namely, MPSs, TTNs and multiscale entanglement renormalization *Ansätze* (MERAs) as quantum circuits, and compared the results with their classical counterparts for one-dimensional data embedding. For simplicity, the quantum circuits with TN-inspired constraints on their gate structure will be referred to as quantum TNs, and their source of inspiration will be classical TNs.[1] Our results have shown that TNs require exponentially more trainable parameters with increasing qubit structure to achieve the same performances as their quantum counterparts, leading to computationally expensive network architectures for machine learning applications. We observed that the classical TNs require exponentially large bond dimensions to capture the same entanglement structure as the QTNs, leaving the stochastic gradient descent (SGD) methods inefficient for optimizing the tensor nodes. This has been improved by employing more extensive Hilbert-space mapping of the input features, indicating that classical TNs require more information to represent the same data as well as QTNs. We present detailed numerical results to study quantum-mechanical differences between TNs and QTNs. In the following, we investigate possible avenues for hybrid classical-quantum TN architectures, allowing a more extensive phase space to be used in the network.

This paper is organized as follows: in Sec. II we introduce TNs and QTNs as methods to perform machine learning tasks. The results have been discussed in Sec. III, where we first introduce the dataset and preprocessing in Sec. III A and then numerical analyses are presented in Secs. III B 1 and III B 2.

II. BRIDGE BETWEEN CLASSICAL AND QUANTUM MACHINE LEARNING

Tensors are multidimensional objects that describe multilinear relations between algebraic entities defined on a particular vector space. This paper employs "tensor diagram notation" to formalize the relation between tensors where a scalar is shown as a node (shown as a blue circle throughout this paper), and each rank is shown with an external line to a given node [40–42].

TNs are defined as a series of Einstein summations indicating the connections between tensor nodes, forming a graph of tensors. Unlike traditional graph networks, however, where each connection indicates the "coupling strength" between nodes, connections between TNs suggest the correlation between a node and the rest of the network and limit the range of entanglement between nodes. These connections between tensor nodes are called bond (or auxiliary) dimensions, shown as χ. Due to the computational cost of contraction of a TN, one can only form specific architectures of TNs that can be contracted efficiently.

As mentioned earlier, TNs are mainly designed to study many-body quantum systems [43] where one-dimensional lattice systems have been extensively studied, and many efficient architectures and contraction algorithms have been developed. A wave function for a one-dimensional lattice with N number of states can be written as

$$|\Psi\rangle = \sum_{\phi_1,\ldots,\phi_n \in \{0,1\}} \mathcal{W}_{\phi_1\cdots\phi_n} |\phi_1\rangle \otimes \cdots \otimes |\phi_n\rangle, \tag{1}$$

where $|\phi_i\rangle$ are spin states spanning a two-dimensional local Hilbert space with an N-state lattice. The subscript of each state identifies the state location within the lattice starting from the first state labeled as 1 till the nth state. $\mathcal{W}_{\phi_1\cdots\phi_n}$ is a rank-N amplitude tensor indicating the bond structure between each state. The top of Fig. 1 shows $|\Psi\rangle$ in tensor diagram notation where each Hilbert-space dimension has been shown by green lines (for simplicity, only seven of them are shown in the image). Each green line represents a two-dimensional vector for a lattice of spin states where outer products form $|\Psi\rangle$. The computational complexity of simulating such objects for spin states is $\mathcal{O}(2^N)$, which grows exponentially with each additional state. However, it is possible to decompose this monstrous beast in smaller tensors that can efficiently represent the original tensor whilst reducing the computational cost of the object.[2]

It is, perhaps, the most intuitive to decompose a one-dimensional lattice into a MPS [26,41,42,46–52], where each state is entangled to the rest of the lattice through the adjacent states.[3] This allows a highly accurate approximation of $|\Psi\rangle$ with only $\mathcal{O}[\text{poly}(N)]$ parameters. Although this limits the entanglement range, the MPS is a potent tool for simulating locally entangled states. The right branch of the classical portion of Fig. 1 shows the representation of MPSs in tensor diagram notation where each blue node represents a rank-3 tensor except for the ones on the edges, forming only rank-2 tensors. By utilizing periodic boundary conditions, one can write a circular network where all nodes are rank-3 tensors. As mentioned before, green lines represent the Hilbert-space dimensions where each blue tensor is connected to a state, $|\phi_i\rangle$, on the lattice. Red lines between tensors represent the auxiliary dimensions, χ, where the size of χ determines the precision of the network by resizing the influence of each state to the rest of the lattice. Although a large bond dimension allows a more precise state representation, it also increases the computational cost of this representation. Hence, different architectures have been proposed to simulate more richly entangled lattice simulations.

Hierarchical TNs can be employed to capture a relatively more complex correlation structure. The TTN is one of the widely used architectures in this avenue [27].[4] TTNs are

[1] It is essential to note that this does not imply that TNs are classical or quantum; the naming choice has been made to simplify the flow of the paper.

[2] Various tensor decomposition methods can be employed to achieve this, such as singular value decomposition [44,45].

[3] For a detailed analysis on MPSs in the context of HEP and machine learning, see Ref. [19].

[4] For a detailed analysis on TTNs in the context of HEP and machine learning, see Ref. [30].

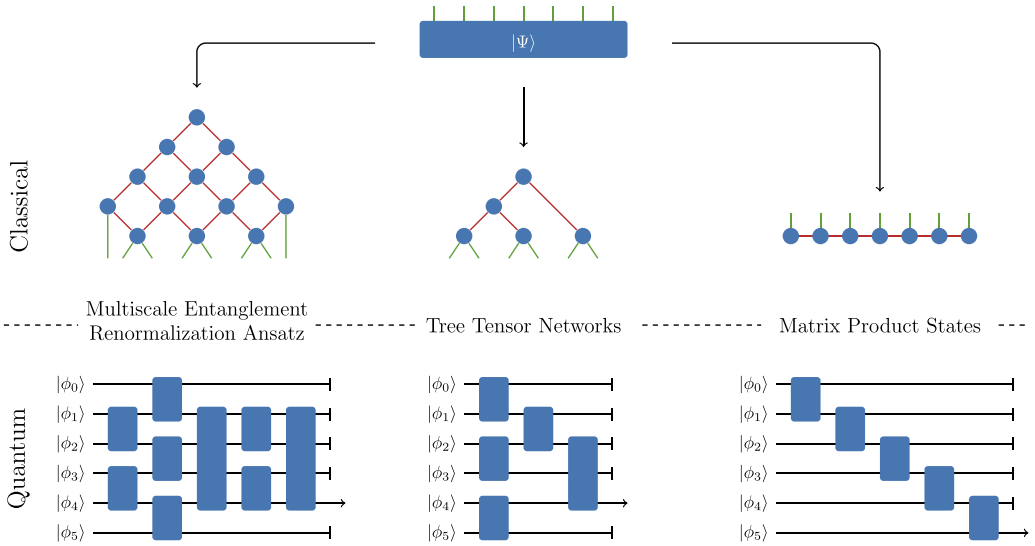

FIG. 1. The upper panel shows the decomposition of the wave function into various classical tensor network realizations. Here the green edges represent Hilbert-space dimensions, and red lines represent auxiliary dimensions between tensors. The bottom panel shows their projection on quantum circuits and each tensor block has been represented as a certain transformation matrix between two qubits. Finally the arrow indicates a measurement. Images are only representative; varied number of inputs are given to simplify the interpretation of the network.

constructed with feature condensing nodes where two or more vectors are condensed into a single vector. This allows neighboring states to be mapped into a higher-dimensional representation at each step. In ML terminology, each node can be seen as a local pooling layer. The middle branch of the classical portion of Fig. 1 shows TTNs in tensor diagram notation. Each green line shows the physical dimensions where neighboring states are collected into one node and then mapped into a higher-dimensional vector shown as red lines, the bond dimensions. Such a hierarchical structure allows each state to have more extensive entanglement allowing them to be entangled with further states on a higher-dimensional manifold. An MPS, in this picture, can be classified as a maximally antisymmetrized TTN.

Further down the rabbit hole, MERAs can be used to embed arbitrarily large entanglement structures into the network [28]. The left branch of Fig. 1 shows a MERA in tensor diagram notation. Although the figure shows a mixture of rank-4 transformation and rank-3 condenser nodes, the MERA does not necessarily need condenser nodes; here, to simplify its application for classification, we mixed the MERA with a TTN. A different MERA architecture for classification can be found in Ref. [53], where after several transformation layers with rank-4 tensors authors reduced the dimensionality via an MPS layer. Each rank-4 tensor plays the role of transformation. In quantum field theory, such nodes have been employed to embed specific known symmetries into the network [54]. Although such a network embodies much higher entanglement between states than MPSs and TTNs, the computational cost is much higher due to the loops that it forms within the architecture.

Tensor networks exhibit similarities with NNs [55,56] where NN layers can be represented as TN layers, leading to more efficient and compressed network structures with similar or better outcomes [57]. TN architectures and quantum-inspired training algorithms can also be independently used as optimization *Ansätze*. In particular, specialized MPS training techniques can be used for classification tasks to achieve similar results to the state-of-the-art NN results [58–62]. The TTN has also been used for quantum assisted classification problems [63–65] and the MERA has been studied as a symmetry embedding layer to a classifier [53,66]. Beyond optimization problems, TNs can improve our understanding of training procedures and network structure due to their widely studied theoretical foundation [67].

For a given ML application, TNs require the feature space of a given datum to be mapped onto a spin state lattice. An N-dimensional feature space, $\mathbf{x} \in \mathbb{R}^N$, can be written as outer products of m-dimensional vectors via a mapping function defined as $\phi(x) := \forall x \in \mathbb{R} \to \mathbb{C}^m$, where for a spin state $m = 2$ [19,58]. Hence the feature space takes the following form:

$$\Phi^{\phi_1 \cdots \phi_n}(\mathbf{x}) = \bigotimes_{i=1}^{N} \phi_i(x_i), \quad \phi_i(x_i) = \sum_{j=0}^{m-1} \alpha_j(x_i)|j\rangle. \tag{2}$$

Here each state is written as a superposition of spin states, $\phi_i(x_i) \in \mathbb{C}^m$, with α_j being feature dependent coefficients, forming an N-dimensional Hilbert space. The correlation between each state, in this form, is controlled by the amplitude tensor, $\mathcal{W}_{\phi_1 \cdots \phi_n}$, given in Eq. (1). The amplitude tensor represents the architecture of the TN, and depending on the decomposition sequence it can be written as any TN architecture; for example, see Fig. 1.

In a classification task, TNs produce the output probability

$$p(\mathbf{x}^{(i)}; \theta) = |f^l(\mathbf{x}^{(i)})|^2 = \left|\mathcal{W}^l{}_{\phi_1 \cdots \phi_n}(\theta)\, \Phi^{\phi_1 \cdots \phi_n}(\mathbf{x}^{(i)})\right|^2, \tag{3}$$

where $f^l(\mathbf{x}^{(i)})$ stands for the contraction of the network with the given ith datum and l represents the output label $\mathcal{W}$ represented as a function of θ, which are the trainable parameters of

the network.[5] Traditional ML applications require each layer to be embedded into an activation function such as ReLu or sigmoid to capture the nonlinear properties of a given datum. However, TNs can capture the nonlinearity without any activation function, ensuring an utterly linear network structure. $\mathcal{W}^{l}{}_{\phi_1\cdots\phi_n}$ consist of trainable parameters where each parameter, θ_j, can be optimized with respect to a given loss function, $\mathcal{L}(\cdot,\cdot)$:

$$\arg\min_{\theta_i\in\mathcal{W}} \mathcal{L}(q(\mathbf{x}^{(i)}),\ p(\mathbf{x}^{(i)};\theta)). \tag{4}$$

Here $q(\mathbf{x}^{(i)}) = y^{\text{truth}}$ represents the truth label of a given ith datum. The representation of the statistical data is the main topic of any ML application.

Entanglement entropy, so-called von Neumann entropy, can be employed as a metric to evaluate the expressiveness of TN states.[6] For a bipartite system of TNs, $\mathcal{I}$, the entanglement entropy is defined as

$$\mathcal{S}(\rho) = -\text{Tr}[\rho\log_2\rho],$$

where ρ is the reduced density matrix of the system $\mathcal{I}$. In a quantum many-body application a nondegenerate pure ground state, $\rho = |\Psi\rangle\langle\Psi|$, is expected to have vanishing entanglement entropy [69,70]. The maximum entanglement entropy of an MPS, on the other hand, is limited by its bond dimension, χ, hence limiting its maximum entanglement entropy to be proportional to $\mathcal{S}(\rho)\sim\ln\chi$. Thus, the required number of parameters in an MPS to represent a quantum system is exponentially large.

Similarly, the MERA has been designed to have the intrinsic property of volume law where its maximum entanglement entropy is bounded by $\mathcal{S}(\rho)\sim s(\mathcal{I})\chi$. Here, $s(\mathcal{I})$ denotes the surface area of a TN graph [68]. Whilst both realizations provide an efficient approximation of a highly entangled system, each has a particular limitation that can significantly increase the computational complexity of the network for an ML application.

Although the entanglement entropy represents the potential of a given *Ansatz* in terms of representation of the underlying data, the trainability of a given model is also an important aspect. In order to efficiently use gradient-based methods, the optimization landscape is required not to be flat. One of the biggest challenges of quantum machine learning (QML) applications is the barren plateau where the gradient of the loss function is exponentially suppressed; hence it is highly challenging to train the model using gradient-based methods [71]. However, the Fisher information matrix can be used to quantify the trainability of a given *Ansatz* which shows the information gained by a given parametrization *Ansatz* [72–74]. For a given probability distribution in Eq. (3), the mean Fisher information is calculated as the variance of the partial derivative with respect to the model parameters, θ, of the log likelihood

$$\bar{F}(\theta) = \frac{1}{|\mathbf{X}|}\sum_{x\in\mathbf{X}}\partial_\theta\log_2 p(x;\theta)\ \partial_\theta\log_2 p^{\top}(x;\theta),$$

where $\mathbf{X}$ is the training sample set. $\bar{F}(\theta)\in\mathbb{R}^{d\times d}$, for d parameters, forms a Riemannian metric capturing the sensitivity of the *Ansatz* with respect to the change in the parameters. In a classical network, the eigenvalue distribution of the normalized $\bar{F}(\theta)$ is mostly degenerate around zero with rare large values [75]. Such distribution shows that the *Ansatz* is not sensitive to the change of most of the parameters.

Based on Fisher information, Ref. [74] has introduced a measure of normalized effective dimensions

$$\hat{d}_{\text{eff}} = \frac{2\log_2\left[\frac{1}{V_\Theta}\int_\Theta\sqrt{\det\left(\mathbb{1}+\frac{|\mathbf{X}|}{2\pi\log_2|\mathbf{X}|}\hat{F}(\theta)\right)}d\theta\right]}{d\log_2\left(\frac{|\mathbf{X}|}{2\pi\log_2|\mathbf{X}|}\right)},$$

where $\hat{F}(\theta)$ is the normalized Fisher information matrix. V_Θ is the volume of the parameter space $\theta_i\in\Theta$. At the $|\mathbf{X}|\to\infty$ limit, $\hat{d}_{\text{eff}}$ of any given *Ansatz* converges to 1, but the convergence of an *Ansatz* is slowed down by small and uneven eigenvalues of the normalized Fisher matrix.

A quantum algorithm is designed through a network of quantum gates to compute specific tasks. The availability of QML methods triggers the question of whether there is a viable quantum gate architecture to harness quantum advantage for ML applications. TNs can be represented as quantum circuits and hence can pose as a viable option for a VQC [32]. Due to the theory built to understand the training and the network structure of TNs, it poses as a powerful variational circuit option for ML applications [31].

An MPS-inspired quantum variational circuit (Q-MPS) can be written by applying a set of unitary transformations to the initial adjacent two-qubit system. Each following two-qubit transformation block takes the last output qubit from a previous block and entangles it to the following qubit. For a six-qubit input, a Q-MPS construction is shown on the right branch of the bottom panel of the Fig. 1. Each blue transformation block represents two unitary transformations, as shown in Fig. 2, followed by a controlled-NOT (CNOT) gate to entangle two states where a generic unitary transformation and the CNOT gate are expressed as

$$U(\theta,\varphi,\lambda) = \begin{pmatrix}\cos(\theta/2) & -e^{i\lambda}\sin(\theta/2)\\ e^{i\varphi}\sin(\theta/2) & e^{i(\lambda+\varphi)}\cos(\theta/2)\end{pmatrix},$$

$$\text{CNOT} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}. \tag{5}$$

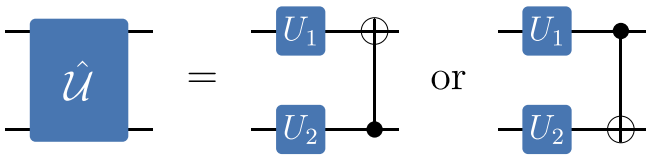

FIG. 2. Representation of a tensor block in a quantum circuit. Each tensor block involves a unitary transformation, $U(\theta,\varphi,\lambda)$, followed by a CNOT gate.

[5]Note that the Einstein-summation convention has been employed throughout the paper.

[6]For a review of entanglement entropy and area law in TNs see Ref. [68], and for entanglement entropy in the context of quantum computing see Ref. [36].

Here θ, φ, and λ are trainable variables. This definition forms the minimal Q-MPS construction. The block structure can also be enhanced via multiple CNOT gates, or auxiliary qubits can be introduced between circuit blocks to increase the bond dimension. Additionally, blocks allowing more qubits have also been proposed in other studies (e.g., see Ref. [32]). To get the classification output, one needs to measure the expectation value of the last entangled qubit:

$$\mathcal{M}_\theta[\Phi^{\phi_1\cdots\phi_n}(\mathbf{x})] = \langle\Phi|\,\hat{\mathcal{U}}^\dagger_{\mathrm{QC}}[U_i(\theta_j)]\,\hat{\mathcal{M}}\,\hat{\mathcal{U}}_{\mathrm{QC}}[U_i(\theta_j)]\,|\Phi\rangle, \tag{6}$$

where $\hat{\mathcal{U}}_{\mathrm{QC}}[U_i(\theta_j)]$ is the given QC constructed by unitarities, $U_i(\theta_j)$, with a set of free parameters, θ_j, and $\hat{\mathcal{M}}$ is a single-qubit operator which we will choose as the third Pauli matrix, $\hat{\mathcal{M}} = \hat{\sigma}_z$. As before, for a classification task, one can define the probability of the network output as the mod square of the expectation value defined in Eq. (6),

$$p(\mathbf{x}^{(i)};\theta) = |\mathcal{M}_\theta[\Phi^{\phi_1\cdots\phi_n}(\mathbf{x}^{(i)})]|^2, \tag{7}$$

and trainable parameters of the system, θ, can be optimised by minimizing Eq. (4).

Similarly, a TTN-inspired quantum variational circuit (Q-TTN) can be constructed by applying a unitary block to each set of adjacent qubits. One can then discard one of the qubits and apply another unitary block to each remaining adjacent qubit. By repeating this process, one can connect the qubits in a hierarchical structure to perform a measure on the top-level qubit. The middle branch of the bottom panel of Fig. 1 shows a six-qubit representation of the Q-TTN.

The MERA-inspired quantum variational circuit (Q-MERA) is closely related to the Q-TTN, but a set of additional unitaries has enhanced the network ahead of each Q-TTN layer. These additional layers allow the circuit to capture more enhanced correlations between qubits that have a similar role in the classical MERA.

Each of the QC representations given in Fig. 1 shows an initial state of $|\phi_i\rangle^{\otimes N}$. However, the initial state of quantum hardware is $|0\rangle^{\otimes N}$; hence a data encoding process has to take place. This can be achieved in two ways: The data can be encoded in each individual qubit amplitude (qubit encoding) or in the entangled state amplitude (amplitude encoding) [76]. In this paper, we will use qubit encoding by rotating each qubit on the y axis with respect to the input values where the rotation is defined as $R_y(x) = U(x, 0, 0)$, as defined in Eq. (5).

In the following section, we will demonstrate the usage of these architectures in the context of top tagging against a QCD background. We have proposed two types of network *Ansatz* where, first, we will employ a purely TN-inspired VQC, and then we will introduce a hybrid *Ansatz* where the data are processed by a classical TN before passing them to the VQC for classification.

III. TOP TAGGING THROUGH QUANTUM TENSOR NETWORKS

The nature of electroweak symmetry breaking (EWSB) has yet to unfold. Due to the sizable corresponding production cross sections, currently, there are billions of top quarks produced at the LHC. With improved capabilities at high luminosity and high-energy LHC in the coming years, the top physics will move to an even higher differential precision era. The large mass of the top quark gives it a unique property of high coupling to the Higgs boson. Accessing the top quark's properties and coupling strength can bring us closer to understanding the nature of EWSB. However, the measurement of the top quark's properties has been obscured mainly due to high levels of background originating from QCD radiation, limiting the analyses to relatively cleaner leptonic final states of the top production. This significantly reduces the production cross section, hence the sensitivity to its properties. In turn, this led the HEP community to investigate the internal structure of jet collimated sprays of radiation [77], where various analytical reconstruction techniques have been developed to understand the substructure of jets originating from different particles. This enabled a sophisticated, precise theoretical understanding of the plethora of highly complex data which cannot be found in any field of science. With the dawn of the deep learning era, these attempts have been shifted towards data-driven analyses.

At the LHC, particularly ATLAS and CMS experiments, the hadronic decay channels of the top quark have been widely investigated. The so-called jet objects, including the information for hadronic activity, deposit their energy in the electronic and hadronic calorimeters in these experiments. These calorimeters can be interpreted as a pixelated cylindrical camera recording the transverse momentum of the radiation shower.

As a demonstration of the capabilities of these networks, we choose to study the discrimination of top quarks against QCD background images reconstructed from energy deposits of the constituents in the electromagnetic and hadronic calorimeters in ATLAS experiment. Calorimeter images provide a natural correlation between pixels, making entangled states highly advantageous for classification processes. Additionally, CNN architectures built to classify jet images have been shown to be highly successful in discriminating gluons from quarks or tops from QCD backgrounds. In this paper, we use these images by mapping them onto a relevant form for the given tensor network (quantum and classical) to process and classify. The code implementation of this study can be found in Ref. [78].

A. Dataset and preprocessing

In this paper, the usage of QTN classifiers has been demonstrated via the dataset provided in Refs. [79,80], often used for benchmarking classification algorithms. It contains collider events for hadronic tops and QCD jets at $\sqrt{s} = 14$ TeV. Following the event generation and parton shower in PYTHIA 8 [81], the detector simulation has been implemented by means of the DELPHES 3 package [82] using its default ATLAS configuration card. In order to capture collimated boosted top topology, jets are reconstructed via the anti-kT algorithm [83] with $R = 0.8$ embedded in the FASTJET [84] package. These jets are also required to be within $p_T \in [550, 650]$ GeV and bounded by $|\eta| < 2$. Top jets are tagged via parton matching using jet radius as the boundary for the angular separation between the jet and the parton level top quark. The dataset

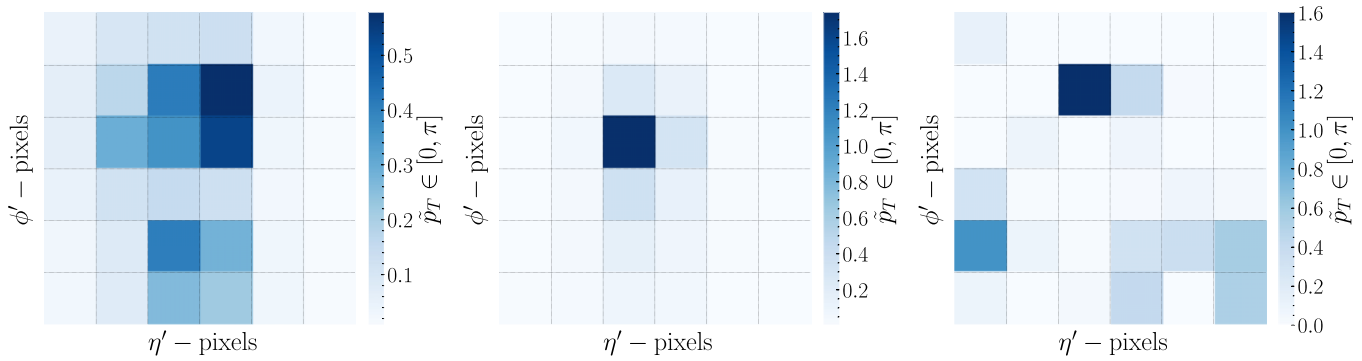

FIG. 3. The left panel shows an average of 5000 top events, and similarly the middle panel shows the same for the QCD background. The right panel shows the top signal for a single event. All images are reduced by cropping 12 pixels from each side and downsampling by averaging four adjacent pixels into one. The color bar shows the intensity of the modified p_T deposited in each reduced pixel which has been fitted within $[0, \pi]$.

includes 1.2×10^6 training and 400 000 separate test and validation samples, respectively.

The jet images are prepared and standardized by following the prescription given in Refs. [19,85] where the constituents of the leading jet, defined above, have been centered on a p_T weighted centroid in the η-ϕ plane. All the modified constituents of the leading jets have been mapped into a 37×37 pixelated frame in the η-ϕ plane spanning within the $[-1.5, 1.5]$ range. Finally, the most energetic image quadrant has been moved to the top right by horizontally and vertically flipping the image. All training samples have been standardized by fitting the pixel values within $[0, \pi]$ range for 200 000 mixed-signal and background samples. Figure 3 shows the standardized average of 5000 events for the top signal (left panel) and QCD background (middle panel). Additionally, the right panel presents a single top signal event. The image has been prepared by cropping 12 pixels from each side and downsampling it by averaging four pixels into one, reducing the image size to 6×6 without losing any vital information.

In order to feed the data into the classical and quantum TNs, the data have to be processed a step further. Since each initial state in a quantum circuit is $|0\rangle$, one needs to prepare this initial state to represent the given data point. Whilst there are various ways of mapping the data on a quantum circuit (see Ref. [76]), we chose to map it by rotating each state around the y axis by the corresponding pixel value using $R_y(\tilde{p}_T^i) = U(\tilde{p}_T^i, 0, 0)$ with $U(\theta, \phi, \lambda)$ defined in Eq. (5). Similarly, as shown in Ref. [58], classical TNs require D-dimensional mapping of the data to simulate the Hilbert space. Such mapping can be performed by using

$$\phi^i(\tilde{p}_T^i) = \sqrt{\binom{D-1}{d_j-1}} \cos^{D-d_j}\left(\tilde{p}_T^i \frac{\pi}{2}\right) \sin^{d_j-1}\left(\tilde{p}_T^i \frac{\pi}{2}\right), \quad d_j \in \{1, \ldots, D\}, \tag{8}$$

where for $D = 2$, $\phi^i(\tilde{p}_T^i)$ reduces to $[\cos(\tilde{p}_T^i \pi/2),\ \sin(\tilde{p}_T^i \pi/2)]$.

B. Network architecture and training

In this section, we will demonstrate a comprehensive numerical analysis and comparison between QTNs and TNs with different sized networks and architectures along with possible hybrid implementations. Our framework relies on TENSORFLOW (version 2.7.0) [86,87] where quantum circuits are simulated using PENNYLANE (version 0.20.0) [88] with QISKIT (version 0.32.1) backend [89] (PENNYLANE-QISKIT plugin version 0.20.0).

1. Classical vs quantum tensor networks

To compare the usage of different architectures presented in the previous section, we prepared two sets of *Ansätze*, one for four qubits and another for six qubits. Such small feature space is owed to the currently accessible quantum hardware limitations, here IBM's quantum hardware. Hence, to employ an extended feature space, we also provide a hybrid classical-quantum TN where the classical portion of the network is responsible for mapping the image into a lower-dimensional manifold, which can then be deposited into the quantum hardware.

To limit the feature space for four and six qubits, the calorimeter image of 37×37 pixels, presented in Sec. III A, has been cropped from each side by 14 pixels and then downsampled, leaving an image of 4×4 pixels. We used the central four pixels for four-qubit networks. For six-pixel networks, we added the adjacent two top pixels of the image. As prescribed in Sec. III A, the modified transverse momentum in each of these pixels is then fitted within $[0, \pi]$. In the following, the image has been reshaped to form a vector.

Figures 4–6 show the four-qubit construction of classical TNs (on the left) and QTNs (on the right) for MPS, TTN, and MERA, respectively. Each TN is required to have bond dimensions of $\chi = 5$ (shown via red lines), and the physical dimensions (shown via green lines) are set to $D = 2$ via Eq. (8). Note that both bond and physical dimensions of the

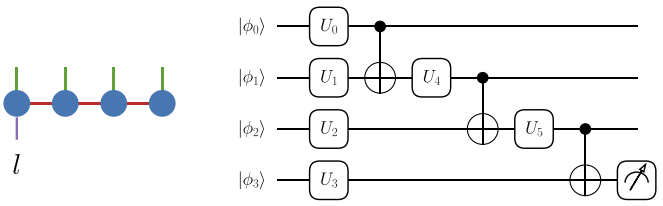

FIG. 4. Representation of the four-qubit classical MPS on the left and Q-MPS on the right.

TNs can be set to much larger values, but we will limit them to compare with the QTN performance. The purple line in each TN shows the prediction output.

Each QTN has been constructed via a set of unitary blocks, which includes trainable parameters where we will limit each unitary block to $U_i(\theta_i, 0, 0)$ by fixing φ and λ to zero. As seen in Figs. 4 and 5 both MPS and TTN networks possess six unitary transformations giving each six trainable parameters. The Q-MERA, shown in Fig. 6, on the other hand, possesses eight trainable parameters.

Similarly, Figs. 7 and 8 show the six-qubit configuration for TNs (on the left) and QTNs (on the right) for TTN and MERA, respectively. Due to its monotonic architecture, an image for the MPS has not been included, but the six-qubit version can be visualized by adding two more qubits to Fig. 4 with the same pattern. The Q-MPS and Q-TTN have nine trainable parameters in this configuration, whereas the Q-MERA possesses 17 trainable parameters due to its complex structure.

For the training of TNs, we employed the Adam [90] optimization algorithm with a learning rate of 10^{-4}. Although it has been noted in a previous study that training TNs with normalized gradients leads to much more stable tensor evolution [19], we will employ standard gradient descent for the TN training. For the four-qubit sample, we used a batch size of 100 events; however, for the six-qubit configuration, this batch size led to an unstable training sequence and hence reduced to 50 events per batch.

For QTNs, we employed quantum natural gradient descent (QNGD) [91,92], which improves the convergence speed in the training of variational quantum circuits [16], for faster optimization with a learning rate of 10^{-2}. Instead of directly updating the given trainable parameters via their gradients for a given loss function, this algorithm solves a linear equation, $\mathbf{M}\lambda = \mathbf{G}$ for λ, where $\mathbf{M}$ is the metric tensor of a given circuit, and $\mathbf{G}$ is the gradient tensor. Then the trainable parameters are updated via $\tilde{\theta}_i = \theta_i - \eta\lambda_i$, where θ_i are the trainable parameters and η is the learning rate. The training has been limited to

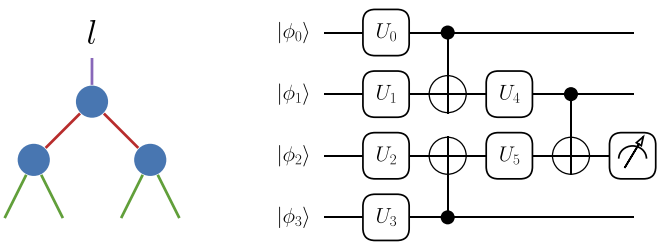

FIG. 5. Representation of the four-qubit classical TTN on the left and Q-TTN on the right.

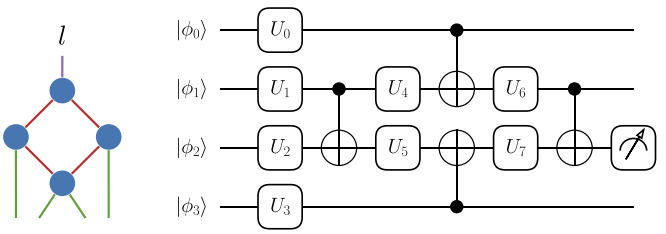

FIG. 6. Representation of the four-qubit classical MERA on the left and Q-MERA on the right.

100 events per batch, and both classical and quantum *Ansätze* have been trained with the full training sample.

Each *Ansatz* has been trained with the cross-entropy loss function,

$$\mathcal{L} = -\frac{1}{N}\sum_{x\in\mathbf{X}} y^{\text{truth}} \log\left[p(x;\theta)\right], \tag{9}$$

where both TNs and QTNs are assumed to be Born machine and hence $p(x;\theta)$ given in Eqs. (3) and (7), respectively. During the training of each *Ansatz*, we observed that loss value evolution for TNs is much slower than QTNs to converge to a minimum loss value; hence TNs are trained for 200 epochs, where QTNs are only allowed to train for 100 epochs. We required the training to be terminated if there is no improvement in validation loss value for 50 iterations, resulting in QTNs only training for 50–55 epochs depending on the *Ansatz*; however, all TN *Ansätze* have run for the entire 200 epochs. We find that QTN training could be terminated within the first 20–25 epochs for both four- and six-qubit configurations. In addition to the early termination condition, we also required learning rate decay for every 25 epoch by a factor of 0.5, depending on the improvement of the validation loss value.

Figure 9 shows a receiver operating characteristic (ROC) curve for four-qubit test results where we only used 10 000 events from the test set due to hardware limitations. For the comparison with the QTN, the bond dimension of each TN *Ansatz* has been chosen to be 5, and the Hilbert-space dimensions are set to 2. Although each wire carries a two-dimensional vector, we observed that $\chi = 2$ led to significantly worse results than the QTN's for any given TN *Ansatz*. The top panel in Fig. 9 shows the results that are executed with the quantum hardware, and the bottom panel shows the quantum simulation execution with various noise models based on

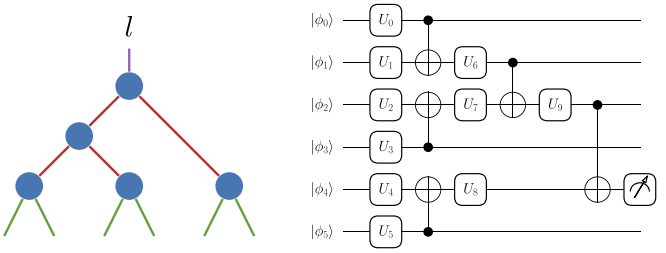

FIG. 7. Representation of the six-qubit classical TTN on the left and Q-TTN on the right.

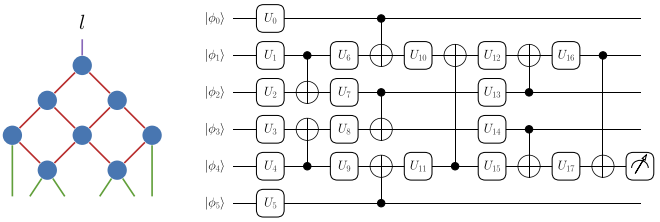

FIG. 8. Representation of the six-qubit classical MERA on the left and Q-MERA on the right.

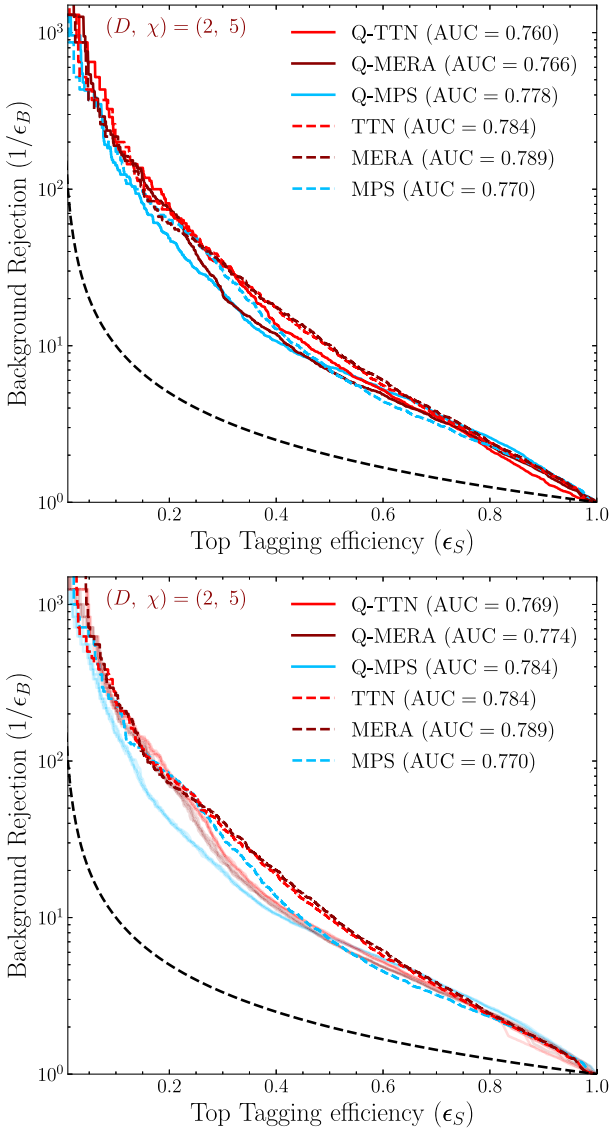

FIG. 9. ROC curve for four-qubit analysis where TTN, MERA, and MPS are shown with red, dark red, and light blue, respectively. Both plots compare classical TNs with quantum versions where QTNs are shown with a solid, dash-dotted, and dotted line for TTN, MERA, and MPS, respectively, where TNs are shown with dashed lines. The top panel shows the comparison where the QTNs are executed in IBM Quantum hardware (ibmq_quito), where the bottom panel shows the same for AER simulation. Multiple shaded lines for AER simulation represent different noise model assumptions. Each sample includes the same randomly chosen 10 000 events from the test sample, and the black dashed line shows a random guess.

five different hardware configurations within IBM Quantum.[7] The same color has visualized the results from each noise model with shaded lines for each *Ansatz*. Each QTN has been executed 5000 times (so-called shots), where the final result is chosen to be the mean of 5000 executions. In both panels, TTNs, MERAs, and MPSs have been represented with red, dark red, and light blue lines, where QTNs are shown with solid, dotted, and dash-dotted lines, and TNs have been shown with dashed lines. For the given four-qubit configuration with $(D, \chi) = (2, 5)$ TTNs, MPSs, and MERAs have 90, 130, and 250 trainable parameters, with their quantum counterparts 6, 6, and 8 trainable parameters, respectively. We have also investigated more generic gate configurations with $U(\theta, \varphi, \lambda)$; however, we did not observe a significant improvement in the results. We observe that both TN and QTN lead to similar results for each other. In addition to the ROC curve, each *Ansatz* has been presented with the area under the curve (AUC) value. For both types of *Ansatz*, we observe minimal change between corresponding realizations. This shows that for a small network configuration, $(D, \chi) = (2, 5)$ is sufficient to represent a similar entanglement structure as QTNs.

Similarly, Fig. 10 shows the ROC curve for a six-qubit configuration with the same color scheme. Again, the top panel shows the results executed with quantum hardware, and the bottom panel shows the quantum simulation results with five different noise models. For both quantum hardware and simulation, we observed that QTN performance had increased by around 17%. Although all QTN architectures lead to similar results for a relatively smaller feature space, we observed significant changes between different QTN realizations after adding two more qubits. While for the four-qubit configuration the maximum difference between AUC values of QTNs was 2.4%; this increased to 3.2% for the six-qubit configuration. Whilst $(D, \chi) = (2, 5)$ leads to comparable results between TN and QTN architectures for four-qubit configurations, we observed that once the network size is increased, with such low bond dimension, TNs were not able to reproduce the same performance as QTNs. For this reason, Fig. 10 shows TN *Ansätze* with larger D and χ values where we managed to match the QTN performance by employing $(D, \chi) = (10, 20)$ for TTN and $(D, \chi) = (5, 10)$ for MPS and MERA architectures.

As a comparison, Table I shows the values for AUC and the number of trainable parameters for each TN *Ansatz* with different D and χ values. We observed that, especially for MPSs and MERAs, only increasing the bond dimension between nodes does not necessarily improve the performance; on the contrary, it has slowed down the optimization process by exponentially increasing the complexity of the loss landscape. Hence increasing Hilbert-space dimensions along with the bond dimensions has been shown to perform better with relatively fewer trainable parameters. Although we did not observe any overtraining for presented values, either method leads to considerable growth in the computational cost of the

[7]Quantum simulations are based on QISKIT's AER package. The simulated noise models are based on ibmq_belem, ibmq_bogota, ibmq_lima, ibmq_manila, and ibmq_quito processors.

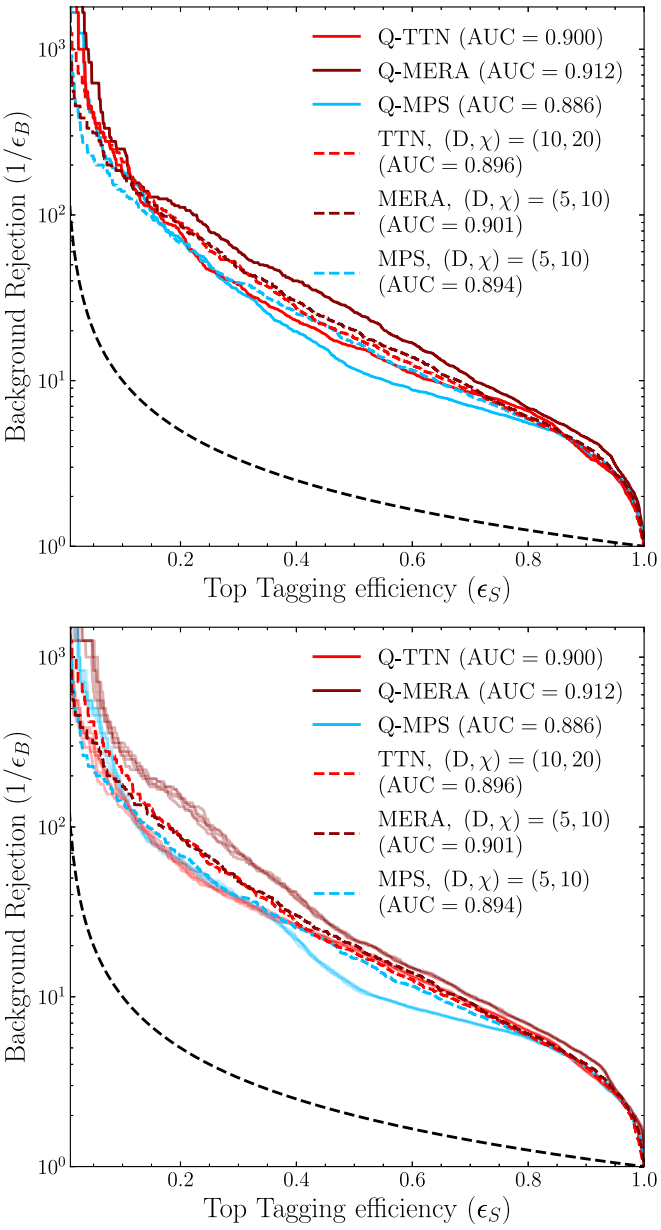

FIG. 10. Same as Fig. 9, showing this time six-qubit arrangement. The top panel shows the samples executed in IBM Quantum hardware (ibmq_perth).

TN contraction, hence resulting in an inefficient network for ML applications.

Although pruned experiments show significant improvement in representability, it is essential to look into a more realistic investigation. For this reason, we employed the full 4×4 image and tested the quality of the classification. Both quantum and classical TNs are reconstructed by simply extending the representations presented above. Instead of using the complete data, however, this time, we used only 10 000 training events for quantum TNs and 50 000 training events for classical networks, which gives a significant disadvantage to the quantum TNs. The rest of the training hyperparameters remained the same as above. Finally, each network has been tested with the same 10 000 test samples as before. For the classical TNs, we used ten-dimensional (10D) Hilbert-space mapping along with 20 bond dimensions, resulting in 64 800 parameters for TTNs, 56 600 parameters for MPSs, and 10 248 004 parameters for MERAs. For their quantum counterparts, on the other hand, we get 30 parameters for Q-TTNs, 30 parameters for Q-MPSs, and 60 parameters for Q-MERAs.

TABLE I. The upper panel of the table shows AUC values and number of trainable parameters for various Hilbert-space (D) and auxiliary (χ) dimensions presented for classical TNs. The bottom panel shows the number of trainable parameters and AUC values for the corresponding QTN *Ansatz*. Each *Ansatz* is designed to receive six pixels (qubits); hence the table provides complementary results for Fig. 10.

Ansatz	D	χ	No. parameters	AUC
TTN	2	5	235	0.755
	2	10	1320	0.803
	2	20	9040	0.849
	5	10	1950	0.873
	10	20	14800	0.896
MPS	2	5	230	0.811
	2	10	860	0.819
	2	20	3320	0.818
	5	10	2150	0.894
MERA	2	5	1225	0.850
	2	10	13400	0.840
	2	20	181600	0.848
	5	10	18200	0.901
Q-TTN	2	2	9	0.893
Q-MPS	2	2	9	0.886
Q-MERA	2	2	17	0.914

Figure 11 shows the false positive rate ratio between classical (ϵ_B^C) and quantum (ϵ_B^Q) networks, plotted against tagging efficiency. From top to bottom, panels show the ratio curve for TTN, MPS, and MERA networks alongside their respective AUC value. For each case, we observe that the performance

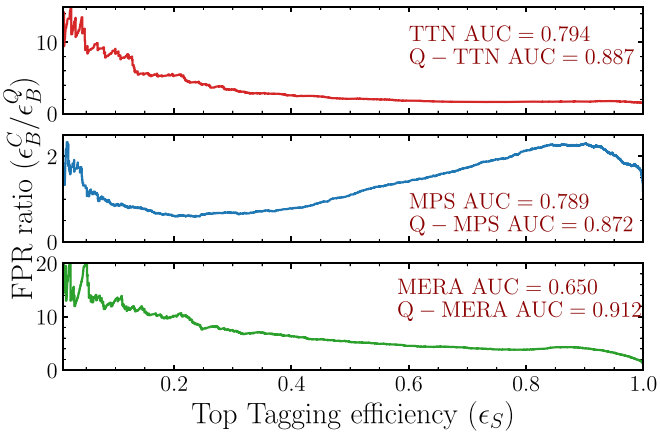

FIG. 11. Background rejection ratio between quantum and classical ROC curves has been presented for 16 feature analyses. From top to bottom, panels show TTN, MPS, and MERA architectures and each panel is presented with a respective AUC value. $(D, \chi) = (10, 20)$ has been used for classical networks.

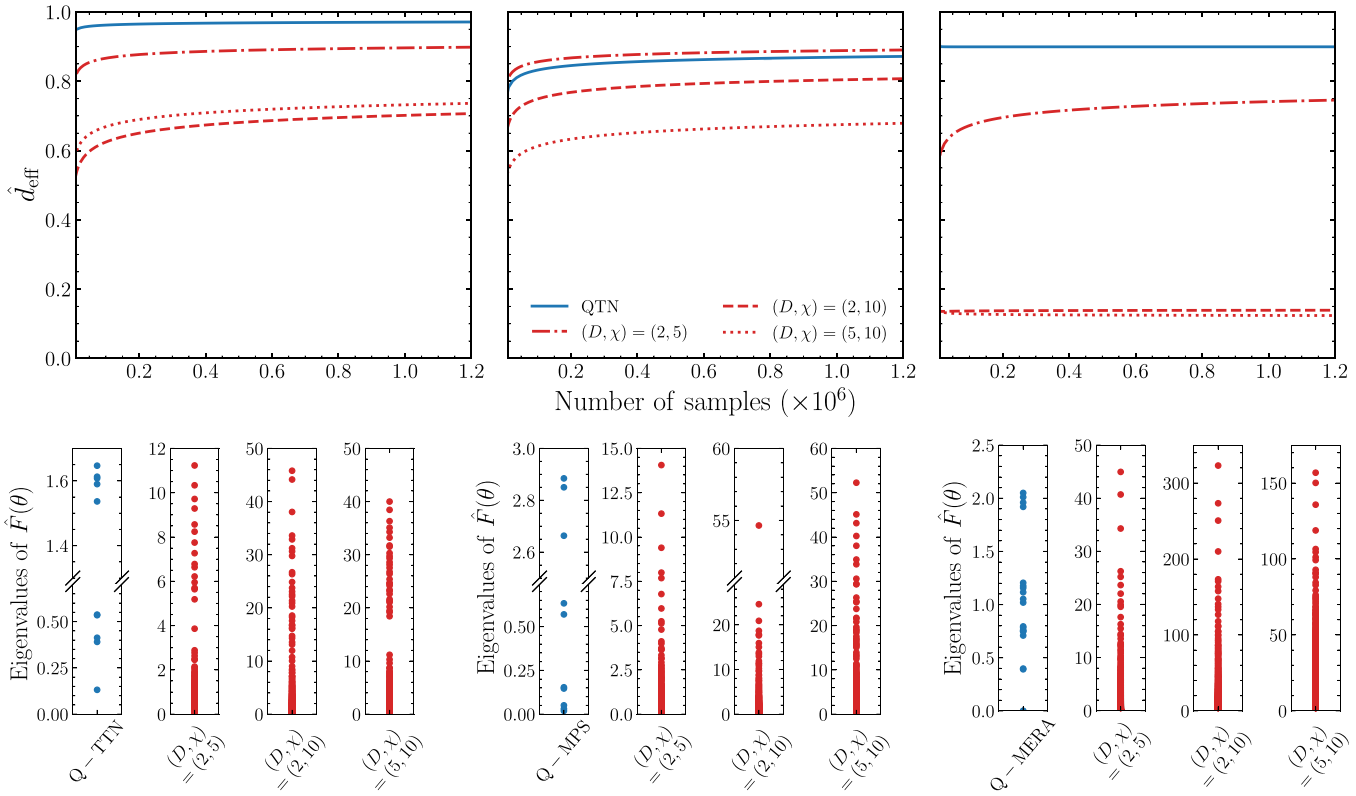

FIG. 12. The upper panel shows the normalized effective dimension with respect to the number of samples for each TN realization, namely, TTN, MPS, and MERA, respectively. QTNs are shown with the solid blue line, and TNs are shown with dot-dashed, dashed, and dotted red lines with respect to corresponding Hilbert-space and bond dimensions shown with (D, χ). The bottom panel shows the eigenvalue distributions of the normalized Fisher matrix for each TN realization with the same color scheme. Again, the plot flow follows from left to right TTN, MPS, and MERA, respectively.

of the QTNs is much superior to the CTNs. It is also essential to note that when trained with only 10 000 samples, CTNs performed significantly worse than the presented outcome. We also observed particularly low variance in the gradients of CTNs, which made the training harder. As before, Q-MERAs showed the best performance among the quantum networks, reaching 0.912 AUC value despite the lack of training samples. Among the classical networks, the MPS seems to achieve the closest result to its quantum counterpart; however, as before, all the classical networks require much larger Hilbert-space mapping and bond dimensions.

It is essential to emphasize that one might naively assume that the claim of Fig. 11 is that the QTN accuracy is higher than its classical counterpart. As noted above, we limited our networks with 10D Hilbert-space mapping with 20 auxiliary dimensions for classical TNs; this can easily be increased to match the accuracy of the QTNs. However, we have limited our network since it is already clear that classical TNs will require significantly larger Hilbert-space mapping and auxiliary dimensions. We also note that we did not observe overtraining; hence it is possible to increase the number of parameters for classical TNs.

In order to study the increase in the complexity of the optimization landscape, we employed the Fisher information matrix presented in Sec. II. The lower three panels of Fig. 12 show the eigenvalue distribution of the normalized Fisher matrix for TTN (left), MPS (middle), and MERA (right). The mean Fisher matrix has been calculated as the average of thousands of executions of randomly chosen input and parameter space. Input values have been uniformly varied within $[0, \pi]$ where trainable parameters have been allowed to run within the $[-\pi, \pi]$ range for each *Ansatz*. Each group shows a blue scatter plot for QTNs and red for different (D, χ) configurations of TNs. We observe that eigenvalue distributions for QTNs are relatively evenly distributed for each case. The Q-MPS shows the most uneven distribution among all other QTN configurations, with more eigenvalues around zero. The eigenvalue distribution for TNs, on the other hand, has been observed to have the typical behavior of a classical network, where the majority of the points are clustered very close to zero, followed by substantially large values.

The effect of the eigenvalue distribution has been reflected in the normalized effective dimension distribution shown in the upper panel of Fig. 12 following the same color scheme, where the (D, χ) configuration has been distinguished by using different line styles. Recall that the normalized effective dimension distribution for each network realization will converge to 1 if a large enough sample has been introduced during the training. We observe that the Q-TTN obtains the most significant effective dimension value, whereas the rest have a more prolonged convergence rate. The effect of eigenvalue distribution can directly be observed in the effective dimension distribution; due to the larger cluster around zero, the Q-MPS shows the lowest values for the effective dimensions.

Whilst a larger TN-bond dimension is essential to have a more accurate representation of the data, we observe that the increase in the bond dimension directly affects the efficiency of gradient-based methods to train the system. The dot-dashed and the dashed lines in Fig. 12 show $(D, \chi) = (2, 5)$ and (2,10). We observe a significant decrease in the effective dimension's convergence rate for each architecture. Similarly, increasing the bond dimension results in a more uneven distribution in the eigenvalues of the Fisher matrix. Comparing these results with Table I, only increasing the bond dimensions does not provide a sufficient increase in the performance of the network, which also sacrifices the trainability of the *Ansatz*. Increasing the Hilbert-space dimension along with the bond dimensions, on the other hand, has been shown to achieve significantly better performance. Table I shows that by only increasing the Hilbert-space dimensions, one can gain around an 8% increase in the AUC values. However, this plummeted the effective dimension convergence rate, leading to a flat optimization landscape.

Despite the significant improvement in the performance, due to the hardware limitations in near-term quantum devices, it is impossible to input a larger η-ϕ plane into a quantum circuit. However, as shown before, since TNs can effectively represent a quantum many-body system instead of manipulating the phase space to be suitable for a small quantum circuit, TNs can be used as a data processing layer which can then be deposited into a quantum circuit for classification. In the following section, such hybrid architectures will be investigated.

2. *Hybrid quantum-classical tensor networks*

This section introduces two main types of hybrid classical-quantum TN architectures. Each architecture is designed to have one classical layer, which will process a large image and reduce it to a four-qubit output. A quantum circuit will classify the processed output from the TN in the following. Since the calorimeter images used in this paper impose two-dimensional correlations between each pixel, we will adjust each given architecture to capture these correlations effectively. We have constrained the architectures to two main groups, namely, one with a TTN and another with an MPS classical layer. Due to the computational complexity, we did not introduce a classical MERA layer that can process the image within a hybrid *Ansatz*.[8]

As mentioned before, TTNs can capture beyond one-dimensional correlations between lattice sites. Concretely, with a more complex TTN tensor structure, it is possible to capture two-dimensional correlations between pixels of an image [65]. We designed a TTN with two types of condenser tensors to achieve this. The tensors connected directly to the image pixels are designed to pool four neighboring pixels, capturing the correlations in both η and ϕ axes. The collection of such tensors can be interpreted as a trainable pooling layer where each node takes four pixels and maps them to a vector on a higher-dimensional manifold. Hence each of these nodes

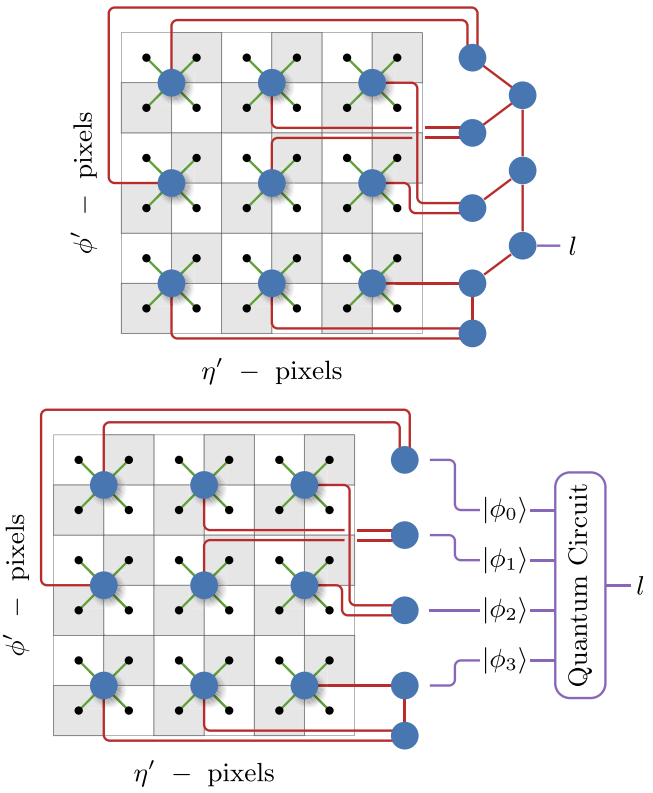

FIG. 13. The left panel shows the representation of the classical TTN designed to capture 2D mapping. The right panel shows the representation of the hybrid classical-quantum TN design where the classical portion is TTN, and the quantum circuit can be any four-qubit variational circuit. Small black circles represent the pixel values, and green lines correspond to Hilbert-space dimensions. Blue circles represent tensor nodes, and the red connections are auxiliary dimensions. Finally, l stands for the network decision.

forms rank-5 tensors. In the following, each of these nodes is connected strategically through rank-3 tensors until we get the desired dimensionality. For a complete classical TTN, the dimensionality is hierarchically reduced until we reach the output dimension. The hybrid realization has been reduced to a concatenated four-dimensional vector for four-qubit input. Figure 13 shows the representation of these two architectures where the network on the top shows the pure classical TN, and the bottom panel shows the hybrid version of the architecture. This specific architecture aims to group the most related sets of tensors to pool the local correlation information before investigating a more global picture. As before, red and green lines represent auxiliary and physical dimensions. The grid represents the image shown in Fig. 3 where the black dots in the center of each pixel represent the mapped pixel vector defined in Eqs. (2) and (8). Purple lines in the bottom panel show the collection of the output values from the classical layer, where each node returns a one-dimensional vector which is then concatenated into a four-dimensional vector to be processed in the quantum circuit. As before, $|\phi_i\rangle$ indicates valid data embedding into the quantum circuit.

Due to the nature of the MPS, its structure is purely limited to one-dimensional lattices. However, as shown in Ref. [19], the image can be reshaped so that the locally correlated pixels are close to each other. An s-shaped reshaping procedure

[8]An application of MERAs on a two-dimensional lattice can be found in Ref. [93].

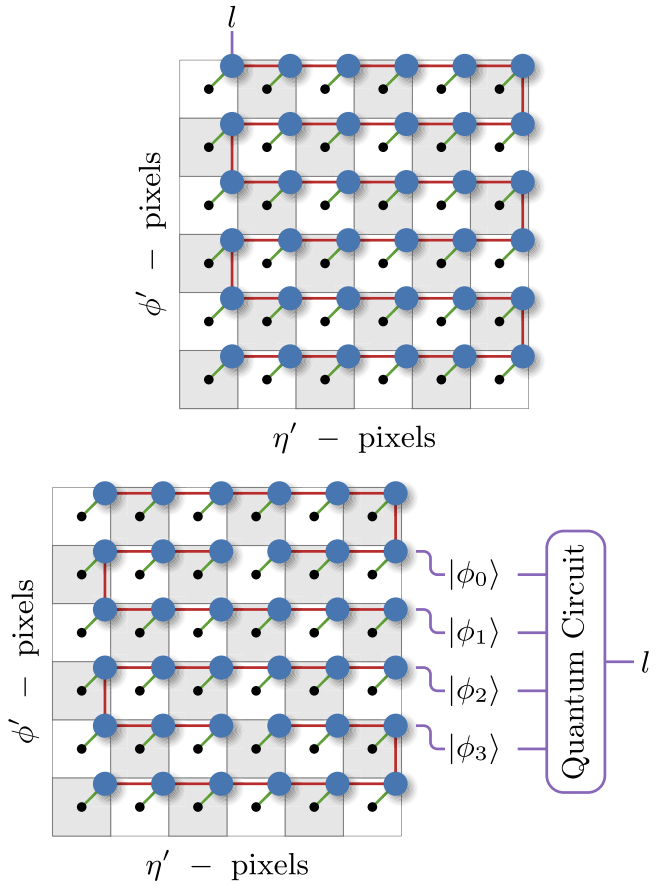

FIG. 14. Representation of hybrid classical-quantum TNs with MPS as a classical layer. The image has been divided into four blocks of nine nodes of MPS, where each MPS output goes into the quantum circuit. As in Fig. 13, green shows Hilbert dimensions and red shows the auxiliary dimensions.

is based on the η axis. Hence, for the classical MPS, we will strictly follow the previously proposed procedure where the pixels are reordered in the η-based s-shaped reordering procedure. For hybrid architecture, on the other hand, the MPS chain has been divided into four blocks of nine nodes where each outputs a one-dimensional vector which then is concatenated and inputted into a quantum circuit. Figure 14 shows the representation of these architectures following the same color scheme as before. The top panel represents the classical MPS, and the bottom panel shows the hybrid architecture.

Such hybrid architecture poses the question of how to train such a network. Although SGD has been proven to be a highly effective training method, it has been repeatedly shown that the QNGD method can achieve much faster convergence for a quantum circuit. Hence we used a mixed optimization algorithm where the classical portion is trained with the Adam algorithm with an initial learning rate of 10^{-4} and the quantum circuit has been optimized via QNGD with an initial learning rate of 10^{-2}. Both learning rates are decayed with the same factor simultaneously, whereas if the loss value of the validation set did not improve for 25 epochs, the learning rate has been reduced by a factor of 0.5. All *Ansätze* are trained with the complete training sample with a batch size of 100 events.

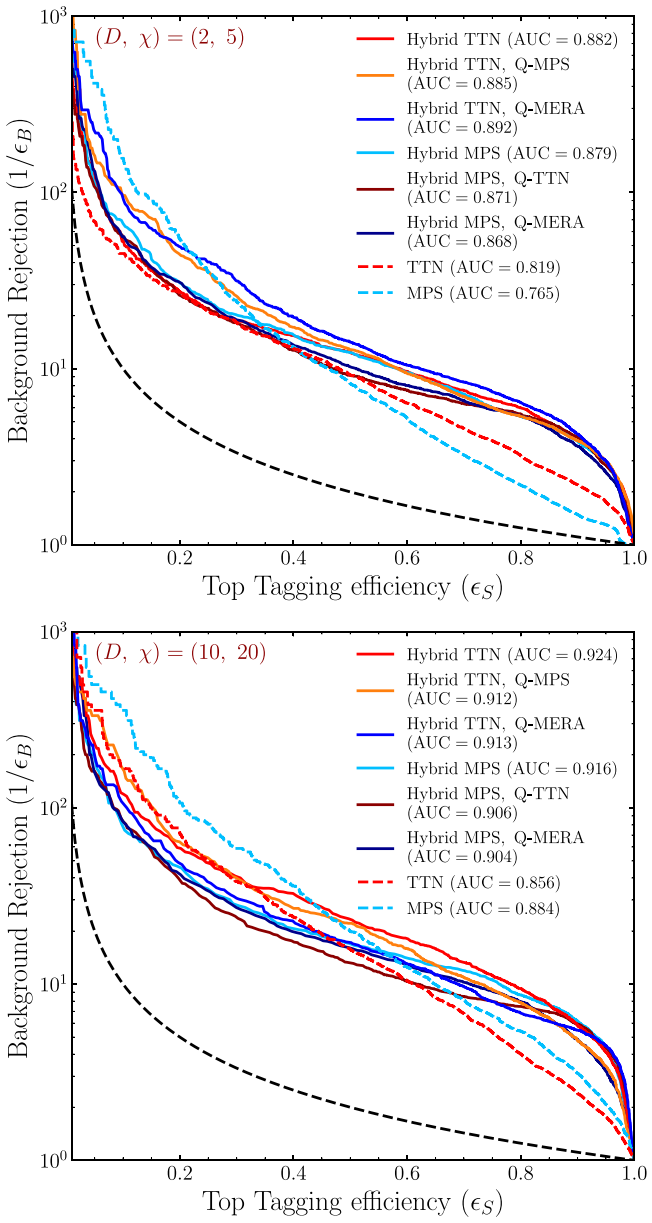

FIG. 15. Same as Fig. 9, showing this time the hybrid classical-quantum TN *Ansatz* with four-qubit arrangement. All quantum circuits have been executed in IBM Quantum hardware (ibmq_quito).

Figure 15 presents the results of various realizations of the hybrid TN architectures compared to purely classical counterparts, where the top panel shows the results generated by setting $(D, \chi) = (2, 5)$. The bottom panel shows the same for $(D, \chi) = (10, 20)$. The dashed red and light blue curves represent both panels' purely classical TTN and MPS realizations. Solid curves represent hybrid *Ansätze* with different QTNs. The architectures with TTNs are shown in red, orange, and dark blue, representing the networks with Q-TTN, Q-MPS and Q-MERA. The light blue, dark red, and dark blue curves, on the other hand, show the MPS with Q-MPS, Q-TTN, and Q-MERA, respectively. Due to the effective two-dimensional representation in the TTN, we observe a 7% increase in the performance compared to the MPS for

the $(D, \chi) = (2, 5)$ configuration. This performance increase mainly originated from the high-efficiency regime, $\epsilon_S \gtrsim 0.4$, but the MPS performs better in the low-efficiency regime. The MPS and TTN possess 1730 and 1645 parameters in this configuration, respectively. However, due to the growth in network complexity, this changes for the $(D, \chi) = (10, 20)$ configuration where the MPS and TTN consist of 136 600 and 1 856 800 parameters, respectively. As before, the TN's optimization capabilities degrade with the complexity of the network, where gradient-based methods are insufficient to train the network efficiently. Hence, although TTN architecture can interpret two-dimensional objects more efficiently than the MPS, it performs worse due to the complexity of the network by a factor of 3%.

We implemented a hybrid test for each classical TN layer using all four-qubit QTNs mentioned in previous sections. As shown with the solid lines, each hybrid realization performs better than the classical versions. However, this improvement is solely based on the high-efficiency regime for each configuration. This might be due to the sensitivity of the networks to the subtle information in the image. The QCD background is mainly concentrated on a few pixels; hence, the information can quickly vanish in a complex network structure if the impact is not significant enough. Whilst for the $(D, \chi) = (2, 5)$ configuration we do not observe a large difference in performance for different hybrid models, the difference gets larger with the $(D, \chi) = (10, 20)$ configuration. Since we used the same number of qubits with the same QTNs, we conclude that this is due to the increase in the mapping capability of the classical layer. For each configuration, the TTN layer has been observed to have a larger impact on the correct classification of the data. However, for the more extensive configuration, the hybrid *Ansatz* with the MPS layer has been observed to achieve much closer results to the TTN.

IV. CONCLUSION

Tensor networks are algebraic tools to represent high-rank tensors effectively. They have been widely studied to represent complex quantum many-body systems and capture their entanglement properties. Multimodal data can be expressed as a quantum system, and TNs can effectively represent correlations within the data structure. Moreover, due to the ability to describe quantum states, TNs are the ideal machinery to study quantum machine learning, where the expertise on "classical" methods built for TNs can directly be applied to quantum hardware.

In this paper, we explored the possibility of classifying HEP data with TN-inspired quantum circuits. We mainly focused on three widely studied TN architectures—TTN, MPS, and MERA—and compared their performance to corresponding quantum circuits. We have shown that, although classical TNs are very successful in representing complex data structures, they require large auxiliary and Hilbert-space dimensions to capture the natural entanglement capabilities of a quantum system. Hence, to achieve the same performance as QTNs, TNs must be executed with a much higher computational cost and more trainable parameters. We additionally observed that TNs require much more training data compared to their quantum counterpart, to be able to sufficiently generalize the output.

The Fisher information matrix provides a Riemannian metric to measure the flatness of the optimization landscape. Based on Fisher information, the effective dimensions relate to the number of samples required to represent the statistical model well. Whilst increasing the network's dimensionality helped improve the performance, using the Fisher information, we observed that this resulted in exponentially suppressed gradients, rendering gradient-based methods unable to train classical TNs efficiently. Additionally, using effective dimensions, we show that TNs require exponentially more data to achieve sufficient representation of the data with increasing auxiliary and Hilbert-space dimensions. Thus, we find that QTNs can perform significantly better than TNs with a fraction of trainable parameters.

Despite the undeniable success of QTNs, they are still constrained to a low number of qubits due to the limitations of quantum hardware in near-term quantum devices, which results in the inability to learn the entire dataset. To surpass this limitation, we proposed a hybrid end-to-end training architecture where a larger phase space of data can be processed with classical TNs and then deposited into a QTN to be classified. Due to the nature of the TNs, this allows a flexible architecture. As a result, more classical nodes can be transformed into circuit inputs once more qubits are available. Finally, we compared the purely classical TNs with hybrid architectures and showed that hybrid networks perform much better than strictly classical TNs.

In this paper, we have limited our QTN architecture to a modest size where each circuit block is transformed into two input states. Although such simple architecture already showed the quantum advantage over classical TNs, these can be extended to blocks transforming more qubits. Additionally, the entanglement between blocks can be enhanced by introducing auxiliary qubits to increase the correlations between circuit blocks. This can significantly improve the expressivity of the quantum network. Additionally, the hybrid architectures presented in this paper were highly simplistic. Since specific geometric properties can be embedded into TN architecture, much more complex structures can be employed using the known symmetries within the data.

ACKNOWLEDGMENTS

We acknowledge the use of IBM Quantum services for this work. In this paper we used ibmq_quito and ibm_perth, which are IBM Quantum Falcon Processors. We thank Vishal S. Ngairangbam and Josh Izaac for very helpful discussions.

[1] B. Nachman, D. Provasoli, W. A. de Jong, and C. W. Bauer, Phys. Rev. Lett. **126**, 062001 (2021).

[2] K. Bepari, S. Malik, M. Spannowsky, and S. Williams, Phys. Rev. D **103**, 076020 (2021).

[3] M. Carena, H. Lamm, Y.-Y. Li, and W. Liu, Phys. Rev. D **104**, 094519 (2021).
[4] K. Bepari, S. Malik, M. Spannowsky, and S. Williams, Phys. Rev. D **106**, 056002 (2022).
[5] T. Li, X. Guo, W. K. Lai, X. Liu, E. Wang, H. Xing, D.-B. Zhang, and S.-L. Zhu (QuNu), Phys. Rev. D **105**, L111502 (2022).
[6] C. Bravo-Prieto, J. Baglio, M. Cè, A. Francis, D. M. Grabowska, and S. Carrazza, Quantum **6**, 777 (2022).
[7] S. Das, A. J. Wildridge, S. B. Vaidya, and A. Jung, Track clustering with a quantum annealer for primary vertex reconstruction at hadron colliders, arXiv:1903.08879.
[8] C. Tüysüz, F. Carminati, B. Demirköz, D. Dobos, F. Fracas, K. Novotny, K. Potamianos, S. Vallecorsa, and J.-R. Vlimant, EPJ Web Conf. **245**, 09013 (2020).
[9] D. Magano, A. Kumar, M. Kālis, A. Locāns, A. Glos, S. Pratapsi, G. Quinta, M. Dimitrijevs, A. Rivošs, P. Bargassa, J. Seixas, A. Ambainis, and Y. Omar, Phys. Rev. D **105**, 076012 (2022).
[10] C. Tüysüz, C. Rieger, K. Novotny, B. Demirköz, D. Dobos, K. Potamianos, S. Vallecorsa, J.-R. Vlimant, and R. Forster, Quantum Mach. Intell. **3**, 29 (2021).
[11] K. Terashi, M. Kaneda, T. Kishimoto, M. Saito, R. Sawada, and J. Tanaka, Comput. Softw. Big Sci. **5**, 2 (2021).
[12] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, Phys. Rev. Res. **4**, 013231 (2022).
[13] S. L. Wu, J. Chan, W. Guan, S. Sun, A. Wang, C. Zhou, M. Livny, F. Carminati, A. Di Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y.-C. Chen, S. Yoo, and T.-C. Wei, J. Phys. G **48**, 125003 (2021).
[14] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, Machine Learning: Science and Technology **2**, 011003 (2021).
[15] A. Blance and M. Spannowsky, J. High Energy Phys. 08 (2021) 170.
[16] A. Blance and M. Spannowsky, J. High Energy Phys. 02 (2021) 212.
[17] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, arXiv:2101.06189.
[18] V. Belis, S. González-Castillo, C. Reissel, S. Vallecorsa, E. F. Combarro, G. Dissertori, and F. Reiter, EPJ Web Conf. **251**, 03070 (2021).
[19] J. Y. Araz and M. Spannowsky, J. High Energy Phys. 08 (2021) 112.
[20] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Proc. R. Soc. A **474**, 20170551 (2018).
[21] R. Sweke, J.-P. Seifert, D. Hangleiter, and J. Eisert, Quantum **5**, 417 (2021).
[22] S. Abel, A. Blance, and M. Spannowsky, Phys. Rev. A **106**, 042607 (2022).
[23] V. S. Ngairangbam, M. Spannowsky, and M. Takeuchi, Phys. Rev. D **105**, 095004 (2022).
[24] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, Phys. Rev. A **101**, 010301(R) (2020).
[25] R. Orús, APS Phys. **1**, 538 (2019).
[26] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, Quantum Info. Comput. **7**, 401 (2007).
[27] Y. Y. Shi, L. M. Duan, and G. Vidal, Phys. Rev. A **74**, 022320 (2006).
[28] G. Vidal, Phys. Rev. Lett. **101**, 110501 (2008).
[29] F. Verstraete, V. Murg, and J. Cirac, Adv. Phys. **57**, 143 (2008).
[30] T. Felser, M. Trenti, L. Sestini, A. Gianelle, D. Zuliani, D. Lucchesi, and S. Montangero, npj Quantum Inf. **7**, 111 (2021).
[31] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, npj Quantum Inf. **4**, 65 (2018).
[32] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, Quantum Sci. Technol. **4**, 024001 (2019).
[33] A. Kardashin, A. Uvarov, and J. Biamonte, Front. Phys. **8**, 586374 (2021).
[34] M. Lazzarin, D. E. Galli, and E. Prati, Phys. Lett. A **434**, 128056 (2022).
[35] A. S. Bhatia, M. K. Saggi, A. Kumar, and S. Jain, Matrix product state based quantum classifier, arXiv:1905.01426.
[36] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, Phys. Rev. Res. **2**, 033125 (2020).
[37] R. Huang, X. Tan, and Q. Xu, Neurocomputing **452**, 89 (2021).
[38] D. Liu, Z. Yao, and Q. Zhang, arXiv:2005.09428.
[39] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, arXiv:2011.14651.
[40] R. Penrose, Combinatorial Mathematics and its Applications **1**, 221 (1971).
[41] R. Orus, Ann. Phys. (NY) **349**, 117 (2014).
[42] J. C. Bridgeman and C. T. Chubb, J. Phys. A **50**, 223001 (2017).
[43] F. Verstraete and J. I. Cirac, arXiv:cond-mat/0407066.
[44] L. Mirsky, Q. J. Math. **11**, 50 (1960).
[45] C. Eckart and G. Young, Psychometrika **1**, 211 (1936).
[46] M. Fannes, B. Nachtergaele, and R. F. Werner, Commun. Math. Phys. **144**, 443 (1992).
[47] A. Klümper, A. Schadschneider, and J. Zittartz, Z. Phys. B **87**, 281 (1992).
[48] I. V. Oseledets, SIAM J. Sci. Comput. **33**, 2295 (2011).
[49] F. Verstraete and J. I. Cirac, Phys. Rev. B **73**, 094423 (2006).
[50] M. B. Hastings, J. Stat. Mech. (2007) P08024.
[51] X. Chen, Z. C. Gu, and X. G. Wen, Phys. Rev. B **82**, 155138 (2010).
[52] U. Schollwöck, Ann. Phys. (NY) **326**, 96 (2011).
[53] J. A. Reyes and E. M. Stoudenmire, IOP Sci. **2**, 035036 (2021).
[54] G. Evenbly and G. Vidal, J. Stat. Phys. **145**, 891 (2011).
[55] N. Cohen and A. Shashua, in *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (2016), Vol. 48, pp. 955–963.
[56] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, Phys. Rev. Lett. **122**, 065301 (2019).
[57] T. Garipov, D. Podoprikhin, A. Novikov, and D. P. Vetrov, arXiv:1611.03214.
[58] E. M. Stoudenmire and D. J. Schwab, Advances in Neural Information Processing Systems **29**, 4799 (2016).
[59] A. Novikov, M. Trofimov, and I. Oseledets, arXiv:1605.03795.
[60] R. Selvan and E. B. Dam, arXiv:2004.10076.
[61] S. Efthymiou, J. Hidary, and S. Leichenauer, arXiv:1906.06329 (2019).
[62] Y. L. Xu, G. G. Calvi, and D. P. Mandic, arXiv:2105.04983.
[63] M. L. Wall and G. D'Aguanno, Phys. Rev. A **104**, 042408 (2021).
[64] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su, and M. Lewenstein, Machine learning by two-dimensional

hierarchical tensor networks: A quantum information theoretic perspective on deep architectures (2018).
[65] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su, and M. Lewenstein, New J. Phys. **21**, 073059 (2019).
[66] F. Kong, X. Yang Liu, and R. Henao, arXiv:2101.03154.
[67] J. Martyn, G. Vidal, C. Roberts, and S. Leichenauer, arXiv:2007.06082.
[68] J. Eisert, M. Cramer, and M. B. Plenio, Rev. Mod. Phys. **82**, 277 (2010).
[69] J. I. Latorre, E. Rico, and G. Vidal, Quantum Info. Comput. **4**, 48 (2004).
[70] M. B. Plenio, J. Eisert, J. Dreissig, and M. Cramer, Phys. Rev. Lett. **94**, 060503 (2005).
[71] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nat. Commun. **9**, 4812 (2018).
[72] J. Martens, Journal of Machine Learning Research **21** (2020).
[73] O. Berezniuk, A. Figalli, R. Ghigliazza, and K. Musaelian, arXiv:2001.10872.
[74] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, Nature Comput. Sci. **1**, 403 (2021).
[75] R. Karakida, S. Akaho, and S.-i. Amari, in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, edited by K. Chaudhuri and M. Sugiyama (IOP Publishing and SISSA, 2019), Vol. 89, pp. 1032–1041.
[76] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, 1st ed. (Springer, New York, 2018), pp. 2364–9062.
[77] S. Marzani, G. Soyez, and M. Spannowsky, *Looking Inside Jets: An Introduction to Jet Substructure and Boosted-Object Phenomenology*, Lecture Notes in Physics Vol. 958 (Springer, Cham, 2019).
[78] https://gitlab.com/jackaraz/tnqcircuits.
[79] G. Kasieczka, T. Plehn, A. Butter, K. Cranmer, D. Debnath, B. M. Dillon, M. Fairbairn, D. A. Faroughy, W. Fedorko, C. Gay, L. Gouskos, J. F. Kamenik, P. T. Komiske, S. Leiss, A. Lister, S. Macaluso, E. M. Metodiev, L. Moore, B. Nachman, K. Nordström, J. Pearkes, H. Qu, Y. Rath, M. Rieger, D. Shih, J. M. Thompson, and S. Varma, SciPost Phys. **7**, 014 (2019).
[80] G. Kasieczka, T. Plehn, J. Thompson, and M. Russel, Top quark tagging reference dataset (2019).
[81] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, Comput. Phys. Commun. **191**, 159 (2015).
[82] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3), J. High Energy Phys. 02 (2014) 057.
[83] M. Cacciari, G. P. Salam, and G. Soyez, J. High Energy Phys. 04 (2008) 063.
[84] M. Cacciari, G. P. Salam, and G. Soyez, Eur. Phys. J. C **72**, 1896 (2012).
[85] J. Y. Araz and M. Spannowsky, J. High Energy Phys. 04 (2021) 296.
[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, software available from tensorflow.org.
[87] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke *et al.*, arXiv:1605.08695.
[88] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, arXiv:1811.04968.
[89] M. S. ANIS, H. Abraham, AduOffei, R. Agarwal, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, S. Anagolum, E. Arbel, A. Asfaw, A. Athalye, A. Avkhadiev, C. Azaustre, P. BHOLE, A. Banerjee, S. Banerjee, W. Bang *et al.*, Qiskit: An open-source framework for quantum computing, 2021.
[90] D. P. Kingma and J. Ba, arXiv:1412.6980.
[91] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, New J. Phys. **21**, 073023 (2019).
[92] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, Quantum **4**, 269 (2020).
[93] L. Cincio, J. Dziarmaga, and M. M. Rams, Phys. Rev. Lett. **100**, 240603 (2008).