Quantum Machine Intelligence (2021) 3: 29
https://doi.org/10.1007/s42484-021-00055-9

RESEARCH ARTICLE

Hybrid quantum classical graph neural networks for particle track reconstruction

Cenk Tüysüz[1] · Carla Rieger[2] · Kristiane Novotny[3] · Bilge Demirköz[1] · Daniel Dobos[3,4] · Karolos Potamianos[3,5] · Sofia Vallecorsa[6] · Jean-Roch Vlimant[7] · Richard Forster[3]

Received: 18 May 2021 / Accepted: 26 October 2021 / Published online: 28 November 2021

Abstract
The Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) will be upgraded to further increase the instantaneous rate of particle collisions (luminosity) and become the High Luminosity LHC (HL-LHC). This increase in luminosity will significantly increase the number of particles interacting with the detector. The interaction of particles with a detector is referred to as "hit". The HL-LHC will yield many more detector hits, which will pose a combinatorial challenge by using reconstruction algorithms to determine particle trajectories from those hits. This work explores the possibility of converting a novel graph neural network model, that can optimally take into account the sparse nature of the tracking detector data and their complex geometry, to a hybrid quantum-classical graph neural network that benefits from using variational quantum layers. We show that this hybrid model can perform similar to the classical approach. Also, we explore parametrized quantum circuits (PQC) with different expressibility and entangling capacities, and compare their training performance in order to quantify the expected benefits. These results can be used to build a future road map to further develop circuit-based hybrid quantum-classical graph neural networks.

Keywords Quantum graph neural networks · Quantum machine learning · Particle track reconstruction

1 Introduction

Particle accelerator experiments aim to understand the nature of particles by colliding groups of particles at high energies and try to observe creation of particles and their decays, e.g. to validate theories. The Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) provides proton-proton collisions to four main experiments as well as other small experiments and fixed-target experiments. In order to achieve a high sensitivity, these experiments use advanced software and hardware.

In addition, these experiments will require very fast processing units as the time between two consecutive collisions is very short (reaching up to 1 MHz for ATLAS and CMS according to The ATLAS Collaboration (2015), Contardo et al. (2015)) and Albrecht et al. (2019). A big data storage and processing problem arise, when the fast data acquisition is combined with sensitive hardware. A total disk and tape spaces of 990 PetaBytes and around 550 thousand CPU cores were pledged to LHC experiments in 2017 according to a report by CERN Computing Resources Scrutiny Group (CRSG) (Lucchesi 2017).

Currently, the LHC is going through an upgrade period to increase the number of particles in the beam (i.e. luminosity) (Apollinari et al. 2015). Therefore, the future High Luminosity LHC (HL-LHC) experiments will require much faster electronics and software to process the increased rate of collisions.

✉ Cenk Tüysüz
cenk.tuysuz@cern.ch

1 Department of Physics, Middle East Technical University, Ankara, Turkey
2 Department of Physics, ETH Zürich, Zürich, Switzerland
3 gluoNNet, Geneva, Switzerland
4 Lancaster University, Lancaster, UK
5 University of Oxford, Oxford, UK
6 CERN, Geneva, Switzerland
7 California Institute of Technology, Pasadena, CA, USA

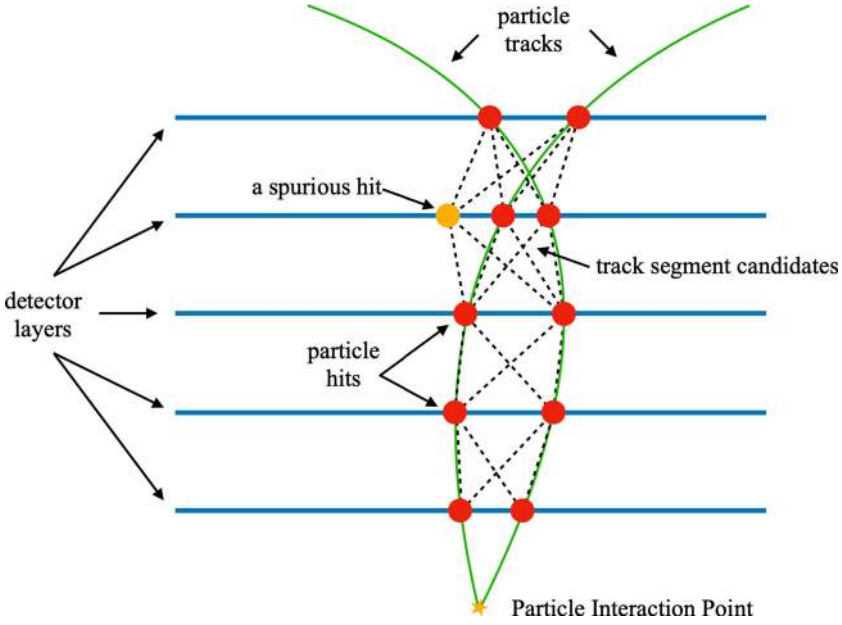


Fig. 1 Drawing of particle track reconstruction problem. Particles interact near the origin of the coordinate system. Products of these interactions travel outwards from the origin. Charged particles bend in a direction depending on their electric charge. When these charged particles pass through the detectors, they create signals called as hits. Particle track reconstruction aims to connect hits belonging to the same particles

In particle track reconstruction problem, the aim is to identify the trajectory of particles using the measurements of the tracking detectors. Accelerated particles interact/collide near the origin of the coordinate system of detectors. Products of these interactions travel outwards from the origin. Charged particles bend in a direction depending on their electric charge. When these particles pass through the detectors, they create signals in the detector called as hits. Particle track reconstruction aims to connect hits belonging to the same particles to assign a trajectory (Fig. 1).

The efficient reconstruction of particle tracks is one of the most important challenges in the HL-LHC upgrade. Although there are novel algorithms (ATLAS Collaboration 2019; Bocci et al. 2020) available that are able to handle the current rate of collisions, they suffer from higher collision rates as they scale worse than quadratically (e.g. $\mathcal{O}(n^6)$) (Magano et al. 2021).

Recent developments in Quantum Computing (QC) allowed scientists to look at computational problems from a new perspective. There is a great effort to make use of these new tools provided by QC to gain high speed-ups for many computational tasks in High Energy Physics (Guan et al. 2021). There are many problems investigated. This include but not limited to physics analysis at LHC using kernel (Wu et al. 2021a; Heredge et al. 2021) and variational methods (Wu et al. 2021b; Terashi et al. 2021),

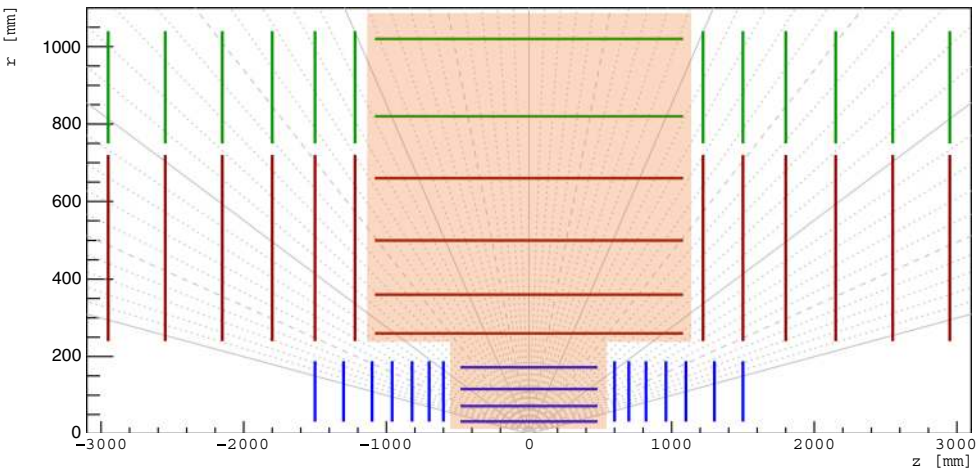


Fig. 2 TrackML detector geometry projected to 2 dimensions (r, z). The region highlighted in orange indicates the detector layers used in this work. Drawing is adapted from Amrouche et al. (2019)

simulating parton showers (Jang et al. 2021) and imitating calorimeter outputs using Quantum Generative Adversarial Networks (Chang et al. 2021).

Researchers have been investigating QC tools for a computational advantage for the particle track reconstruction problem, since it also suffers from scaling. While there are several attempts using adiabatic QC (Bapst et al. 2019; Zlokapa et al. 2019), Quantum Associative Memory Shapoval and Calafiura (2019) or Quantum search routines (Magano et al. 2021), this work focuses on hybrid variational methods.

In this work, we aim to give a complete overview on our developments, where we investigated the use of a hybrid quantum-classical graph neural network (QGNN) approach to solve the particle track reconstruction problem (Tüysüz et al. 2020a, 2020b, 2020c) that has been trained on the publicly available TrackML Challenge dataset (Amrouche et al. 2019, 2021). We present an analysis of several well-performing Quantum Circuits and give a comparison with its classical equivalent, HEP.TrkX (Farrell et al. 2018), on which our approach is based on.

The rest of the paper is organized as follows. Details of the dataset and pre-processing methods are given in Section 2. The QGNN model is explained in detail in Section 3. Results and comparisons with novel methods are given in Section 4, along with a discussion of the findings. Finally, our summary and comments on possible improvements are presented in Section 5.

2 The dataset and pre-processing

The publicly available TrackML Challenge dataset provides 10,000 events to emulate the HL-LHC conditions (Amrouche et al. 2019). It has become a benchmark dataset for researchers after the conclusion of the challenge (Amrouche et al. 2021) and allows comparisons across different methods. The simulated tracking detector geometry of the dataset is that of a general purpose collider experiment. The schema of this geometry in 2 cylindrical coordinates (r,z) can be seen in Fig. 2. Horizontal layers in the center of the detector represent a barrel-shaped geometry, while vertical layers represent a disk-shaped geometry and are generally referred to as end-cap layers.

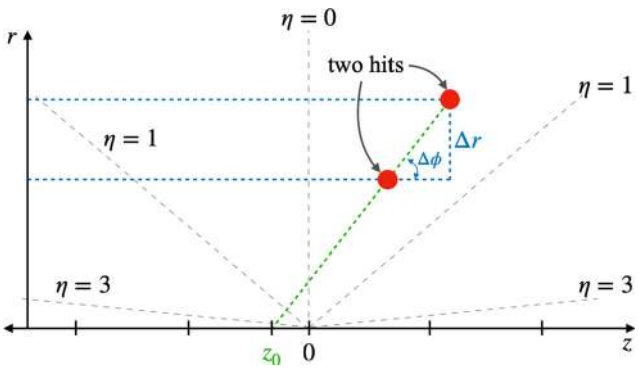


Fig. 3 A sketch of the cylindrical coordinate system for particle collisions. The beam is along the z-axis and the particles collide near z=0. The r axis is the projection of the transverse (x-y) plane

In many Quantum Machine Learning applications, it is very hard to work with large datasets due to restrictions on simulation times. A pre-processing step is necessary, which reduces the amount of samples and prepares the data format for the model.

The pre-processing procedure starts by selecting the first 100 events from the dataset. Although it would be ideal to use all events of the dataset, computation time restrictions of the QC simulation limited us to use only a portion of the dataset. Then, particle hits are restricted to the barrel region of the detector, which is the region highlighted in Fig. 2. This limits the number of tracks and the ambiguity in identifying the particle trajectories. In addition, a cut in the transverse momentum of the particle is applied to further reduce the number tracks.

After reducing the number of tracks to reasonable numbers, the next step is to create graphs out of the remaining particle hits. Particle hits become nodes and the track segment candidates will be defined as edges of graphs at this stage. Then, a set of restrictions is applied to all possible graph edges for creating a graph with as less fake edges as possible, while preserving as many true edges as possible.

These restrictions are defined using a cylindrical coordinate system, which is widely used in High Energy Physics to leverage the symmetries of the detectors. We follow the same convention and present some of the definitions visually in Fig. 3 for further clarification.

As the first step of the hit graph construction, only the edges that connect nodes of consecutive detector layers are considered. Then, edges with the pseudorapidity ($\eta = -\ln(\tan(\phi/2))$ larger than 5 are eliminated. Pseudorapidity is a measure of the angle to the z-axis used in High Energy Physics.

Next, the ratio of difference in ϕ to r ($\Delta\phi/\Delta r$), where ϕ is the angle to the z-axis, is required to be smaller than 6×10^{-4}. Finally, a z intercept (z_0) of all edges is required to be smaller than 200 mm to eliminate highly oblique edges.

The Pseudo-code of the algorithm is also presented in Algorithm 1. Detailed plots of these selections are presented in Appendix A.

Algorithm 1 The pre-processing of an event.

```
procedure PREPARE
    for all hits do
        if hit in Barrel Region then
            keep hit
        end if
    end for
    for all particles do
        if pT > 1 GeV then
            for each hit of particle do
                X[hit_id] = [r, φ, z]
            end for
        end if
    end for
    set Ri, Ro and y = 0
    for hit pairs of consecutive detector layers do
        if η < 5 then
            if Δφ/Δr < 0.0006 and z0 < 100 mm then
                Ri[hit_0_id, segment_id] = 1
                Ro[hit_1_id, segment_id] = 1
                if particle_0_id == particle_1_id then
                    y[segment_id] = 1
                end if
            end if
        end if
    end for
    return X, Ri, Ro, y
end procedure
```

In total, 100 graphs from 100 events are obtained with this method. The graph production is done with a 99% efficiency and a purity of 51%, which are defined as:

$$\text{Efficiency} = \frac{\#\text{ of selected true track segments}}{\#\text{ of initial track segments}}, \tag{1}$$

$$\text{Purity} = \frac{\#\text{ of selected true track segments}}{\#\text{ of selected track segments}}. \tag{2}$$

After the graph construction, the dataset is stored in form of 4 matrices; $X \in \mathbb{R}^{N_V \times 3}$ stores 3 spatial coordinates of all nodes (in cylindrical coordinates; r, ϕ, z), R_i and R_o ($R_i, R_o \in \{0,1\}^{N_V \times N_E}$) store input and output nodes of all edges, and $y \in \{0,1\}^{N_E}$ stores the labels of edges. Their definitions can be seen below.

$$R_i^{jk} = \begin{cases} 1, & \text{if } k^{th} \text{ edge is input of } j^{th} \text{ node} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$R_o^{jk} = \begin{cases} 1, & \text{if } k^{th} \text{ edge is output of } j^{th} \text{ node} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$y_k = \begin{cases} 1, & \text{if nodes of } k^{th} \text{ edge belong to same particle} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Constructed graphs have 8784 $\pm$ 1877 edges (N_E) and 5583 $\pm$ 804 nodes (N_V) on average. An example graph showing the fake and true edges is presented in Fig. 4.

The pre-proprecessing method is identical to the one used in HEP.TrkX project (Farrell et al. 2018), except the pT restriction, which is used to reduce total number of particles in an event. This is done intentionally in order to compare our results with the classical equivalent model.

3 The hybrid quantum-classical graph neural network model

A graph neural network (GNN) is a Neural Network model that acts on features of the graph, such as nodes, edges or global features (Veličković et al. 2018). GNNs have shown great success in many occasions for node and graph classification and link prediction (Wu et al. 2021c). Their success led to applications in High Energy Physics for many problems such as track and particle flow construction (Farrell et al. 2018; Ju et al. 2020; Shlomi et al. 2021; Biscarat et al. 2021; Pata et al. 2021). This situation attracted the interest of the Quantum Machine Learning community to develop quantum graph neural networks for different applications (Verdon et al. 2019; Chen et al. 2021).

The hybrid quantum-classical graph neural network (QGNN) model that we propose takes a graph as the input and returns a probability as the output for all edges of the initial graph. The QGNN builds up an attention passing graph neural network model proposed by Veličković et al. (2018), following the same strategy as the HEP.TrkX project of Farrell et al. (2018). In contrast to the classical GNN approach, we add a Quantum Neural Network (QNN) layer to Multi Layer Perceptrons (MLP).

The QGNN consists of 3 parts. The first one is the Input Network, whose task is to increase the dimension of the input data. It takes the spatial coordinate information (e.g. 3 cylindrical coordinates) and passes them through a single fully connected Neural Network layer with sigmoid activation and an output size corresponding to the hidden dimension size (N_D). Then, these new data points are concatenated ($\oplus$) to form the initial node feature vector, where $v \in \mathbb{R}^{N_V \times (3+N_D)}$.

$$v = x \oplus \phi_{FC}(x) \tag{6}$$

As a next step, the node feature vector is fed to Edge and Node Networks, which process the graph iteratively in order to obtain a final edge probability value (e) for each of the edges. During this process, the same Edge and Node Network is sequentially executed for a predetermined number of iterations (N_I) times and finally the same Edge Network is used one more time to obtain final edge

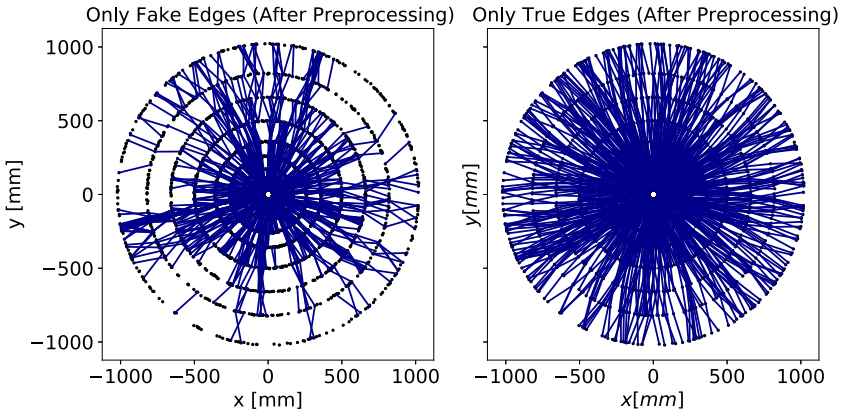


Fig. 4 Graphs are produced with the pre-processing of each event. 2D projection of hits, fake and true edges of an event are plotted. All hits are plotted with black circles. Fake (on the left) and true (on the right) edges of a graph are plotted in the Cartesian coordinates (transverse plane). There are 5162 true and 5508 fake edges of this event

probabilities ($e \in [0,1]^{N_E}$). This pipeline is summarized with a simple drawing in Fig. 5.

3.1 The Edge Network

The Edge Network takes pairs of nodes into account and returns the probability for those two nodes to be connected. Initially, the connectivity of each pair of nodes is given by the connectivity matrices R_i and R_o. Using these matrices, node feature vectors b_o and b_i of all initially connected edges, or so called *doublets* ($b_o \oplus b_i$), are obtained.

$$b_o^k = \sum_{j=1}^{N_V} R_o^{jk} v_j \quad b_i^k = \sum_{j=1}^{N_V} R_i^{jk} v_j \tag{7}$$

The feature vectors of input and output nodes of each edge are concatenated in order to be fed into a Hybrid Neural Network (HNN, $\phi_{EdgeNetwork}$). The HNN returns edges features (e), which are the probabilities for each edge, to be part of a real trajectory or not. Next, the edge features are passed to the Node Network.

$$e_k = \phi_{EdgeNetwork}\left(b_o^k \oplus b_i^k\right) \tag{8}$$

3.2 The Node Network

The Node Network builds up on the edge feature matrix given by its predecessor, the Edge Network. Based on this input information, the node features are updated. In this case, a combination of each node of interest and its neighbors from upper and lower detectors is created, forming a *triplet*. Here, the node features of the neighbors' are scaled with the corresponding edge features.

$$v'_{j,input} = \sum_{k=1}^{N_E} e_k R_i^{jk} b_o^k \quad v'_{j,output} = \sum_{k=1}^{N_E} e_k R_o^{jk} b_i^k \tag{9}$$

Similar to the Edge Network, the triplet is fed to a Hybrid Neural Network ($\phi_{NodeNetwork}$).

$$v_j := \phi_{NodeNetwork}\left(v'_{j,input} \oplus v'_{j,output} \oplus v_j\right) \tag{10}$$

This time, the HNN returns new node features v. The updated features are passed again to the Edge Network and this process is repeated for N_I times. This allows the aggregation of information from farther nodes of the graphs and updates the hidden features accordingly.

3.3 The hybrid neural network

Our approach employs Hybrid Neural Networks (HNNs), which combine both classical and quantum layers. The

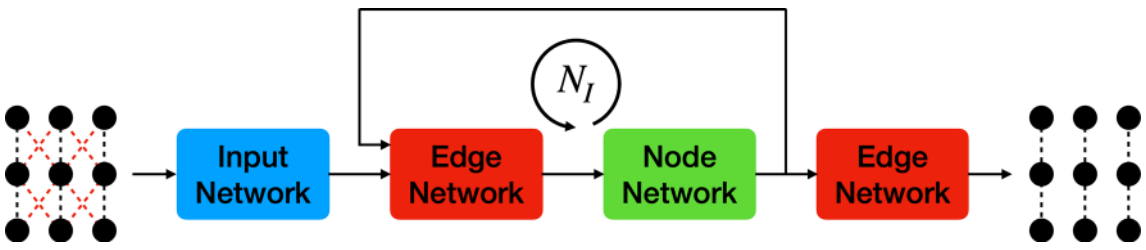


Fig. 5 Schematic of the QGNN architecture. The pre-processed graph is fed to an Input Network, which increases the dimension of the node features. Then, the graph's features are updated with the Edge and Node Networks iteratively, number of iterations (N_I) times. Finally, the same Edge Network is used one more time to extract the edge features of the graph that predicts the track segments. There is only one Edge Network in the pipeline, two Edge Networks are drawn only for visual purposes. The pipeline is adapted from Farrell et al. (2018)

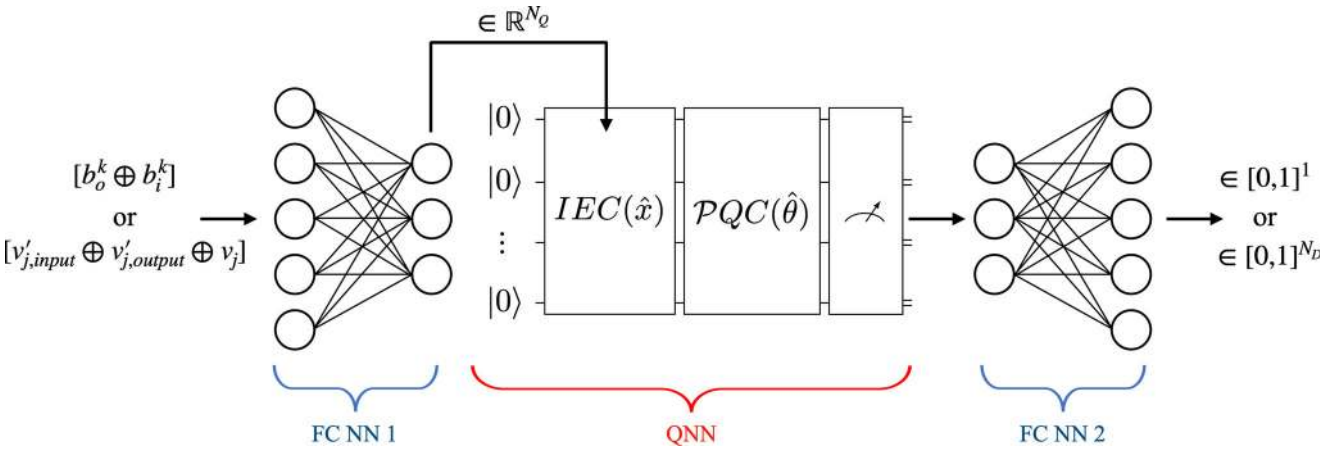


Fig. 6 The Hybrid Neural Network (HNN) architecture. The input is first fed into a classical fully connected Neural Network (FC NN) layer with sigmoid activation. Then, its output is encoded in the QNN with the information encoding circuit (IEC). Next, the parametrized quantum circuit (PQC) applies transformations on the encoded states. The output of QNN is obtained as expectation values for each qubit that is measured. A final FC NN layer with sigmoid activation is used to combine the results of different qubit measurements. The same HNN architecture is used in Edge (upper input and output dimension) and Node Networks (lower input and output dimension) with different parameters. The input and output dimension sizes change according to the network type. Details of the dimensions of each layer are given in Table 1

HNN starts with a single fully connected neural network (FC NN 1) layer with sigmoid activation. The output dimension of this layer is equal to number of qubits (N_Q) used by the quantum layer. Then, the output of the FC NN 1 is used in the encoding step of the QNN. Finally, the measurements of the QNN are fed to another FC NN with sigmoid activation, which has the output dimension of 1 (in the case of Edge Network) or hidden dimension size (N_D) (in the case of Node Network). This architecture, as presented in Fig. 6, allows full flexibility in the hidden dimension size, the number of qubits and the type of the QNN. Details of input and output dimensions of all layers can be seen in Table 1.

The QGNN model is experimented with different quantum layers to understand potential benefits. These type of quantum models with parametrized quantum circuits have been called differently in the literature (McClean et al. 2018; Farhi and Neven 2018; Benedetti et al. 2019; Mitarai et al. 2018; McClean et al. 2016; Romero and Aspuru-Guzik 2021). Here, we use the name Quantum Neural Network (QNN) as we use them in a similar fashion to Neural Network layers.

Table 1 Input and output dimensions of layers used in the HNN. QNN has the output dimension of 1 if the circuit measures only one qubit (e.g. MPS and TNN)

Layer	I/O	Edge Network	Node Network
FC NN 1	Input	$2 \times (3 + N_D)$	$3 \times (3 + N_D)$
	Output	N_Q	N_Q
QNN	Input	N_Q	N_Q
	Output	1 or N_Q	N_Q
FC NN 2	Input	1 or N_Q	N_Q
	Output	1	N_D

QNN has the output dimension of N_Q if all qubits are measured (e.g. Circuit 10 and Circuit 19)

The QNN of our choice consists of three consecutive parts. An information encoding circuit (IEC) encodes classical data to states of the qubits followed by a parametrized quantum circuit (PQC) that is applied to transform these states to their optimal location on the Hilbert Space. Finally, measurements are performed along the z-axis with the σ_z operator.

Information encoding has a significant effect on the training capacity of QNN models (Schuld et al. 2021), therefore a lot of attention is required when deciding on how to do it. We employ angle encoding, because it provides an encoding which uses significantly less gates compared to others, e.g. amplitude encoding, and it needs almost no classical processing (Larose and Coyle 2020).

Encodings such as amplitude encoding allow encoding of classical information by using significantly less qubits, but this advantage is usually reverted by the number of gates required to build the circuit. For example, the amplitude encoding of a feature vector $x \in \mathbb{R}^n$ only uses $log_2\, n$ qubits but needs 4^n single and two qubit gates. On the other hand, the angle encoding requires n number of qubits and a single qubit gate per qubit (Leymann and Barzen 2020). This allows an easier implementation and experimentation with angle encoding. An angle encoding of a four dimensional

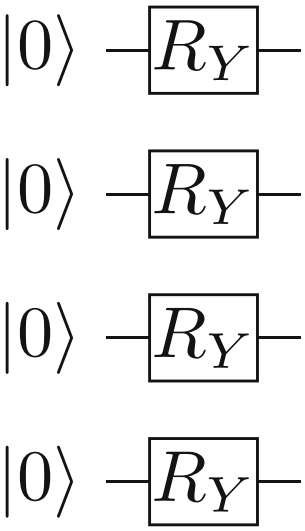


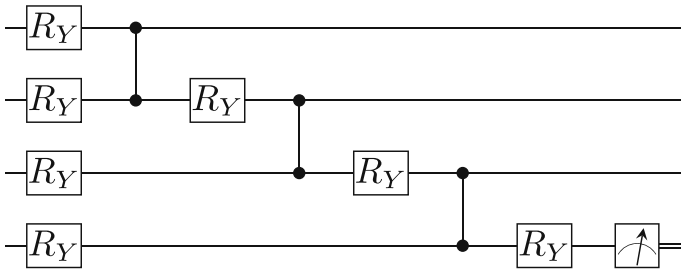
Fig. 7 Angle encoding quantum circuit of a four dimensional feature vector with respect to the y-axis

feature vector can be performed, e.g. with the circuit given in Fig. 7.

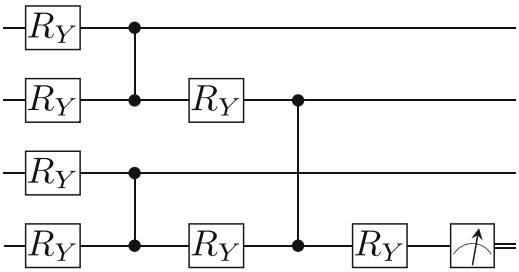
The QNN encodes classical incoming information on the qubits via rotational gates in the desired axis using angle encoding. In order to obtain a unique and bijective representation of the classical data, the rotation angle is mapped between $\theta \in [0, \pi]$ due to the periodicity of the cosine function. This is relevant since the expectation value is taken with respect to the σ_z-operator at the end of the circuit execution.

The PQC is the part of the QNN model that is going to be tuned in order to provide the desired output. As in classical Neural Network layers, those initially randomly assigned variables are optimized during training to fit the certain training objective, i.e. to minimize the overall loss function. In order to achieve a good training performance, choosing a good combination of IEC and PQC is essential. Although there are many practical and theoretical work to understand this better (Sim et al. 2019; Leyton-Ortega et al. 2021; Hubregtsen et al. 2021), our current understanding of which combination works for which task is still limited (Schuld et al. 2021). We therefore try to cover a range of PQCs and fix the IEC to a specific angle encoding to provide more controlled results. We consider two types of PQCs.

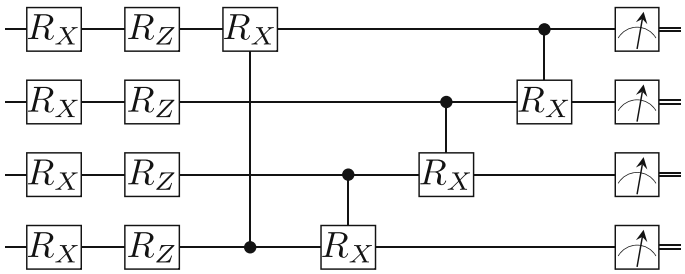
The first PQC type consists of circuits with a hierarchical architecture. Matrix Product State (MPS) (Bhatia et al. 2019) and Tree Tensor Network (TTN) (Grant et al. 2018) inspired circuits belong to this group. However, these PQCs measure only one qubit. Thus, they are only implemented in case of the Edge Network as a multi-dimensional output is needed for the Node Network. Examples of MPS and TTN circuits can be seen in Fig. 8.



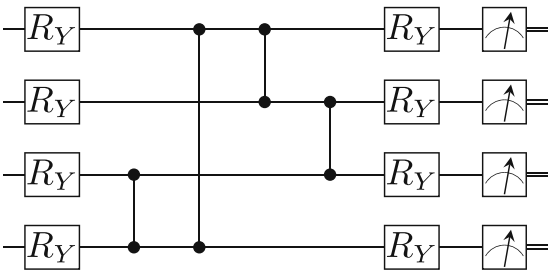
(a) Matrix Product State (MPS) circuit. Adapted from Bhatia et al. (2019).



(b) Tree Tensor Network (TTN) circuit. Adapted from Grant et al. (2018).



(c) Circuit 19 in four qubits and single layer configuration. Adapted from Sim et al. (2019).



(d) Circuit 10 in four qubits and single layer configuration. Adapted from Sim et al. (2019).

Fig. 8 Hierarchical (a, b) and layered (c, d) PQCs used in the HNN. (a) MPS applies two-qubit gates with a ladder-like architecture, (b) while TTN uses a tree-like architecture implement R_Y and CZ gates. MPS and TTN circuits measure only one qubit at the end. (c) Circuit 19 employs R_X, R_Z and CRX gates in a nearest neighbour fashion, (d) while Circuit 10 do this with R_Y and CZ gates. Circuit 10 and Circuit 19 can be extended to any number of layers by repating the circuit. Circuit 10 and Circuit 19 measure all qubits available

Table 2 Labels of PQC settings used in the HNN

Label	Edge Network	Node Network
`circuit 10`	Circuit 10	Circuit 10
`circuit 19`	Circuit 19	Circuit 19
`MPS-10`	MPS	Circuit 10
`TTN-10`	TTN	Circuit 10

The second type of PQCs are more common in the QML literature. They consist of layers of parametrized gates that are generally followed by controlled operations. These circuits act on all qubits fairly, meaning all qubits can be measured to obtain information. This makes them suitable for both Edge and Node Networks. Another important advantage of this type of PQCs is having different descriptors in the literature, which allows to compare different properties such as expressibility and entanglement capacity. In this work, two circuits with different expressibility and entanglement capacity were chosen from Sim et al. (2019), namely Circuit 10 (Fig. 8d) and Circuit 19 (Fig. 8c).

This work compares the two different configurations of PQCs. Models with the labels `circuit 10` and `circuit 19` use the same circuits with different initial parameters for the Edge and Node Networks, as it was done in previous comparisons. While the models with `TTN-10` and `MPS-10` labels use `circuit 10` for the Node Network and either a TTN or an MPS type of PQC for the Edge Network. These definitions are also presented in Table 2.

Expressibility measures a PQC's ability to explore the Hilbert Space (Sim et al. 2019). It is a numerical method that samples two random states from a given PQC. Fidelities of these states are computed and a distribution ($\hat{P}_{PQC}(F;\theta)$) is obtained after collecting many samples (e.g. 5000 samples for four qubits). Then, this process is repeated by sampling Haar random states. Finally, two distributions are compared using Kullback Leibler divergence (D_{KL}). Expressibility (E) is expressed as;

$$\mathrm{E} = D_{KL}(\hat{P}_{PQC}(F;\theta) \parallel P_{Haar}(F)) \tag{11}$$

The value of Expressibility is less for more expressive circuits. In order to avoid confusion $E' = -log_{10}(E)$ will be used as Expressibility. Hence, the Expressibility value (E') increases with more expressive circuits (Hubregtsen et al. 2021).

Similarly, Entanglement Capability is a numerical method that quantifies a PQCs ability to produce entangled states (Sim et al. 2019). It averages the Meyer-Wallach entanglement measure (Q) over many random samples obtained from the PQC (e.g. 5000 samples for four qubits). For example, a fully entangled two qubit state $\left(|\Psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}\right)$ has $Q(|\Psi\rangle) = 1$ and a state with no entanglement (e.g. $|\Phi\rangle = |01\rangle$) has $Q(|\Phi\rangle) = 0$. Entanglement Capability (Ent) is expressed as;

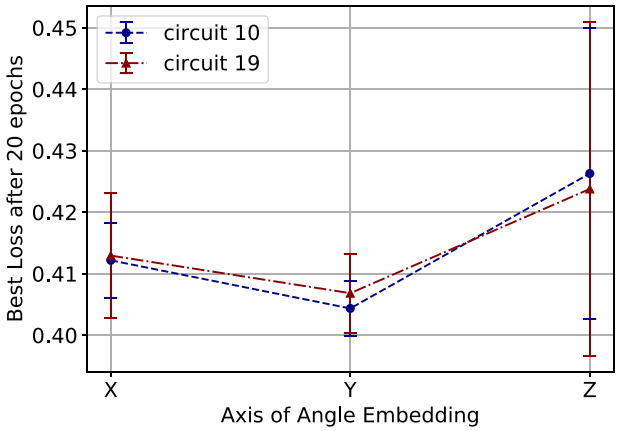
$$Ent = \frac{1}{\|S\|}\sum_{\theta_i \in S} Q(|\psi_{\theta_i}\rangle) \tag{12}$$

Expressibility and Entanglement Capability are descriptors that allow comparison of layered PQCs. Expressibility and Entanglement Capability improve with more layers and will be a point of interest in our discussions. For example, Circuit 10 has less Expressibility and Entanglement Capability compared to Circuit 19, with the same number of qubits and layers. This is because Circuit 10 has only R_Y and CZ gates compared to additional R_Z and CR_X gates of Circuit 19, which brings additional degrees of freedom. The differences between these circuits will be another point of interest in our comparisons to better understand the behaviour of PQCs. Expressibility and Entanglement Capability of Circuit 10 and Circuit 19 is presented in Appendix B.

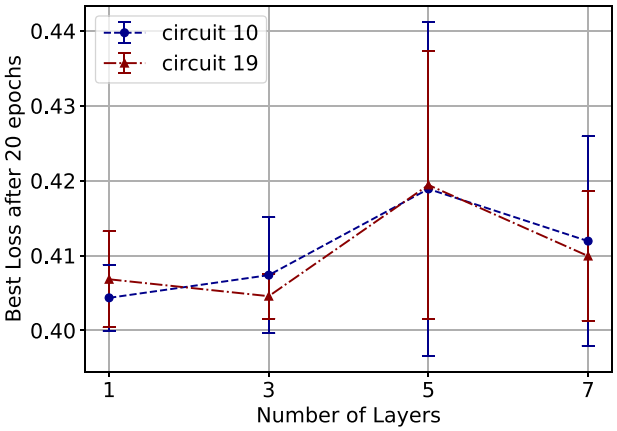
The reason behind analyzing PQCs under two types is related to their response to scaling with increasing number of qubits and layers. We use gradient-based optimizers due to the hybrid nature of QGNN model. Gradient-based optimizers require the model to produce strong enough gradient signals to be able to explore the loss landscape. This might become a problem when a model is scaled. Barren Plateaus are the name given to flattening of the loss landscape (McClean et al. 2018). They appear in models where gradients vanish exponentially with an increasing model size and they are one of the greatest challenges in training variational quantum algorithms (VQAs) (Cerezo et al. 2021). In general, layered-type PQCs suffer from Barren Plateaus, which makes them hard or even impossible to train for a large amount of qubits and layers. On the other hand, the absence of Barren Plateaus were shown for some PQCs with hierarchical architectures, such as Quantum Convolutional Neural Networks (QCNNs) (Pesah et al. 2020) and TTNs (Zhao and Gao 2021). Because of this, comparing these two types of PQCs have great importance to better understand the behaviour of hybrid models at large scales.

3.4 Training the network

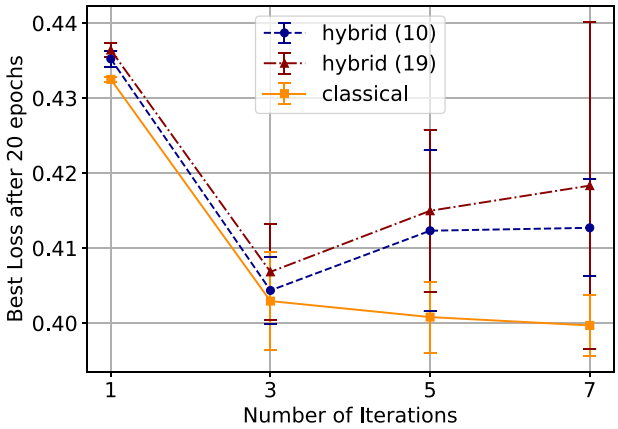
Training hybrid quantum-classical neural networks requires software that can differentiate both types of networks. Pennylane by Bergholm et al. (2018) is one of the most popular open-source tools that provides this feature. Pennylane was used along with PyTorch (Paszke et al. 2019) during the early stages of this work. However, this combination turned out to be too slow to handle both



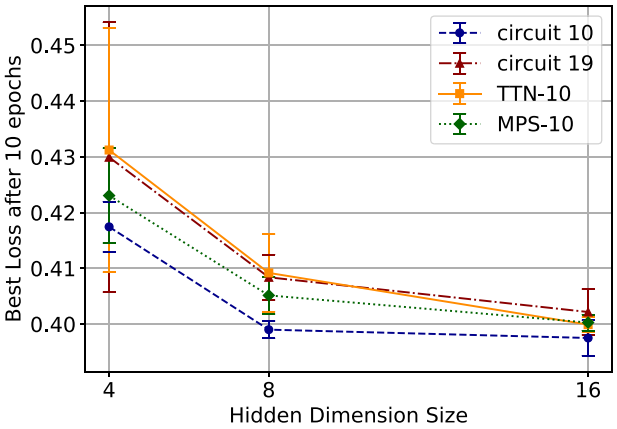
(a) Axis of angle embedding comparison.



(b) Number of layers (N_L) comparison.



(c) Number of iterations (N_I) comparison.



(d) Hidden dimension size ($N_D = N_Q$) comparison.

Fig. 9 Best validation loss comparison with respect to different parameters of the hybrid GNN model. (a) The axis of angle embedding comparison considers the best loss obtained for different embedding axes by setting $N_D = N_Q = 4$, $N_L = 1$ and $N_I = 3$. (b) The number of layers comparison considers the best loss for various numbers of layers (N_L) by setting $N_D = N_Q = 4$ and $N_I = 3$. (c) The number of iterations comparison considers the best loss for different numbers of iterations (N_I) by setting $N_D = N_Q = 4$ and $N_L = 1$. (d) The hidden dimension size comparison considers the loss for different hidden dimension sizes (N_D) by setting $N_Q = N_D$, $N_L = 1$ and $N_I = 3$. 5 instances of all models with different initial parameters are trained for 10 or 20 epochs depending on complexity for each setting, and the mean of best losses are presented. The error bars represent the $\pm$ standard deviation of the best losses of all 5 runs

the dataset and the model. A computational speed-up in training has been achieved using Qulacs (Suzuki et al. 2020). Although it provided faster training, it was still not enough. Finally, the combination of Cirq, Tensorflow and Tensorflow Quantum (Cirq Developers 2021; Abadi et al. 2016; Broughton et al. 2020) produced the optimal scenario, in which we were able to reduce the training times to less than a week. The quantum circuit simulations are performed with only taking analytical results into account, i.e. without sampling the quantum circuits. Although analytical results do not reflect hardware conditions, we made this choice in order to obtain results in a reasonable amount of time.

The 100 events selected from the dataset are separated randomly with a 50/50 ratio to be used as training and validation sets. Models are trained using the binary cross entropy loss function, given in Eq. 13, where y_i is the truth label and $\hat{y}_i$ is the model prediction for an edge.

$$L = -\frac{1}{N_E}\sum_{i=1}^{N_E} y_i \log\left(1-\hat{y}_i\right) + \left(1-y_i\right)\log\,\hat{y}_i \tag{13}$$

The Adam optimizer (Kingma and Ba 2017) with a learning rate of 0.01 is used to train all trainable parameters of the hybrid model. The learning is done with a batch size of 1 per graph and continued up-to 10 or 20 epochs

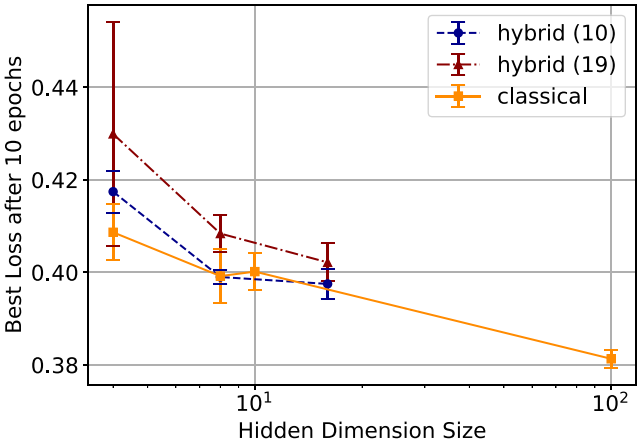


Fig. 10 Best validation loss comparison with respect to different hidden dimension sizes (N_D) of the hybrid and classical GNN models. The comparison is made with the choice of $N_Q = N_D$, $N_I = 3$ and $N_L = 1$. 5 instances of all models with different initial parameters are trained for 10 epochs, and the mean of best losses are presented. The error bars represent the $\pm$ standard deviation of the best losses of all 5 runs

depending on model complexity. All models are trained for 5 independent initializations and their mean is presented in all results.

4 Results and discussion

We trained the hybrid model with many configurations to explore the potential of the method. Here, we present four key comparisons of features that have a significant effect on the performance of the model.

First, the effect of angle embedding axis choice on the training performance of `circuit 10` and `19` is compared. Circuit 10 is a PQC that consists of R_Y and CZ gates, while circuit 19 is a PQC with R_Z, R_X and CRX gates. The comparison is made by setting number of qubits and number of hidden dimension size to 4 ($N_Q = N_D = 4$). Then, number of layers is also set to 1 ($N_L = 1$) and number of iterations is set to 3 ($N_I = 3$). The best loss values of each model is plotted in Fig. 9a. In both cases, the x and y-axis embedding resulted in better loss values, compared to z-axis. The z-axis embedding requires deeper circuits to match other axes' representation capacity, since the measurements are taken with respect to the Pauli-Z operator. Because of this outcome, the y-axis angle embedding is considered for the rest of the results. Training curves of these comparisons are presented in Appendix F.

There are contrasting results in the literature on how expressibility and entanglement capacities affect the training performance. Recently, Hubregtsen et al. (2021) showed a positive correlation between expressibility and accuracy, while Leyton-Ortega et al. (2021) showed the opposite. They found that more expressive models perform worse and also overfit more. On the other hand, entanglement has been shown to limit the trainability of models depending on how it propagates in between qubits by Marrero et al. (2020) and Zhang et al. (2020). To better understand the situation on our case, we tested two models with `circuit 19` and `circuit 10` with various number of layers. `circuit 19` has better expressibility and entanglement capacity compared to `circuit 10` (Sim et al. 2019). Best loss values obtained after a training with 20 epochs is plotted with respect to number of layers in Fig. 9b with $N_Q = N_D = 3$ and $N_I = 3$. We could not observe a significant difference between two models. This situation might be a result of using an encoder and decoder consisting of a fully connected neural network in the model, which could have compensated for the different expressive capacity of the models. However, increasing expressibility and entanglement capacity of both models resulted in worse performance in both cases. In this way, our results are consistent with results of Leyton-Ortega et al. (2021). This behaviour is thought to be the result of Barren Plateau formation (McClean et al. 2018). Training curves of these comparisons are presented in Appendix D.

The number of iterations of a GNN is an important parameter that determines a model's performance (Farrell et al. 2018; Veličković et al. 2018). It allows propagation of information to farther nodes. A comparison with $N_Q = N_D = 3$ and $N_L = 1$ is made with `circuit 10` and `circuit 19` and the results are presented in Fig. 9c. Training results show that the best loss is obtained for $N_I = 3$ for the hybrid cases. However, this is not the case in the classical case. Ju et al. (2020) report an $N_I = 8$ as the optimal value for their model with 128 hidden dimensions in their extended project Exa.TrkX. The increase in the value of the lowest loss with increasing number of iterations might be due to low expressive capacity of the whole model, as this comparison is made only with a $N_D = 4$. Training curves of these comparisons are presented in Appendix E.

In order to investigate how these hybrid models scale, their performances with respect to increasing the hidden dimension size and qubits are compared. This comparison is made with the choice of $N_Q = N_D$, $N_I = 3$ and $N_L = 1$. Two different configurations of PQCs are compared. Models with the labels `circuit 10` and `circuit 19` use the same circuits with different initial parameters for the Edge and Node Networks, as it was done in previous comparisons. While the models with `TTN-10` and `MPS-10`

labels use Circuit 10 for the Node Network, a TTN or an MPS type of PQC for the Edge Network. These definitions are given in Table 2.

A comparison of the best losses is made after 10 epochs and presented in Fig. 9d. The performance of the models improve consistently with the increasing hidden dimension size. This shows that learning capacity of the model benefits from more dimensions. The model with `circuit 10` outperforms the rest consistently. However, there seems to be a saturation of the best loss as the hidden dimension size increases. Training curves of these comparisons are presented in Appendix C.

Finally, the hybrid model is compared against the classical model at different hidden dimension sizes and presented in Fig. 10. For this comparison the same choice of $N_Q = N_D$, $N_I = 3$ and $N_L = 1$ is followed. This result shows that the hybrid model scales similarly to the classical model until a certain hidden dimension size. We did not perform simulations for qubits larger than 16 due to restrictions set by simulation times and classical hardware resources.

5 Conclusion

In this work, we implemented a hybrid quantum-classical GNN (QGNN) model for particle track reconstruction using the TrackML dataset (Amrouche et al. 2019). This is the first end-to-end implementation of a hybrid quantum-classical GNN model with attention passing to the best of our knowledge. We showed that the model can perform similar to classical approaches for low number of hidden dimension sizes. We investigated how the model scales for different hyper-parameters. `circuit 10` consistently performed the best among other models in all comparisons. On the other hand, `circuit 10` has the worst expressibility and is the lowest entangling model in a single layer configuration. Numerical results indicate that larger PQC models are harder to train, as it was shown in many instances (McClean et al. 2018; Leyton-Ortega et al. 2021).

The current status of Quantum hardware restricted us to use only simulations of Quantum Circuits. This was mainly due to thousands of circuit executions required by the model. Because of the high pile-up conditions of the TrackML dataset, the graphs have thousands of nodes and edges, and therefore using hardware is a challenge for this approach. A forward pass of the presented QGNN model builds $N_I + 1$ circuits for each edge and N_I circuits for each node. This also limited us experimenting with larger sized models due to restrictions in simulating Quantum Circuits. In order to cope with this problem, we enforced a pT-cut for reducing the number of particles, a small number of qubits (up to 16) was used. We also used analytical results with no noise and trained models up to 20 epochs at maximum. Very large RAM requirements and the significant increase of training times to more than a week for models with 16 qubits were the limiting factor in our results.

This work explores an advantage in reducing the size of high dimensional NN layers with Quantum Circuits that have significantly less qubits. However, results obtained with $N_Q = N_D$ were only able to match the performance of the classical model. On the other hand, this does not mean that an advantage is impossible. There is more to explore to better understand the potential of this approach.

First, the QGNN model was only experimented with simple encoding circuits (angle encoding), while more sophisticated encodings are conjectured to significantly affect the performance of QNN-based models (Schuld et al. 2021). Furthermore, recent work by Abbas et al. (2021) showed that QNN models with four qubits and ZZ Feature Maps of depth two has a larger effective dimension compared to its classical equivalent, which leads to a better learning capacity. Therefore, a further study is needed to explore the potential benefits of different data encodings.

On top of that, this work does not explore any noise effects, which is considered as one of the limiting factors of VQAs as it can lead to Barren Plateaus (Wang et al. 2021). Hardware noise is conjectured to slow down training of VQAs. However, this does not mean that noise always disfavors VQAs. It is argued that hardware noise can help to explore the loss landscape (Cerezo et al. 2021). In many instances, VQAs were shown to have noise resilience (Sharma et al. 2020; Gentini et al. 2020) and benefit from noise (Cao and Wang 2021; Campos et al. 2021). Furthermore Mari et al. (2021) showed that gradients and higher order of derivatives can be accurately obtained under hardware and shot noise. The results from the literature indicate that understanding the effect of noise on variational models is essential to estimate their potential. In this work, experiments with noise were attempted but then abandoned due to technical limitations posed by the size of the dataset.

The QGNN model can be further improved by employing better training schemes (Leyton-Ortega et al. 2021) and noise aware optimizers (Arrasmith et al. 2020). Further research directions include exploring more sophisticated data encodings and understanding effect of noise. It would be more beneficial to work with a smaller dataset for exploring hybrid GNN models that target NISQ hardware.

Appendix A: The dataset and pre-proceesing details

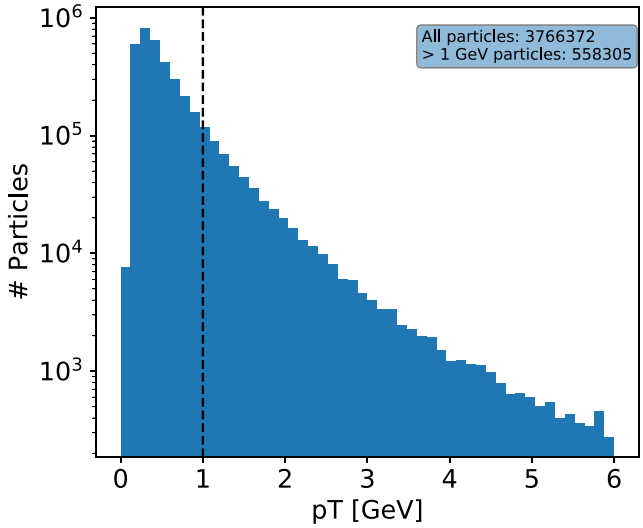


Fig. 11 Histogram of number of particles in 100 events that are inside the barrel region of the TrackML detector. The dashed line represents the 1 GeV pT threshold we have selected in order to reduce total number of particles and tracks

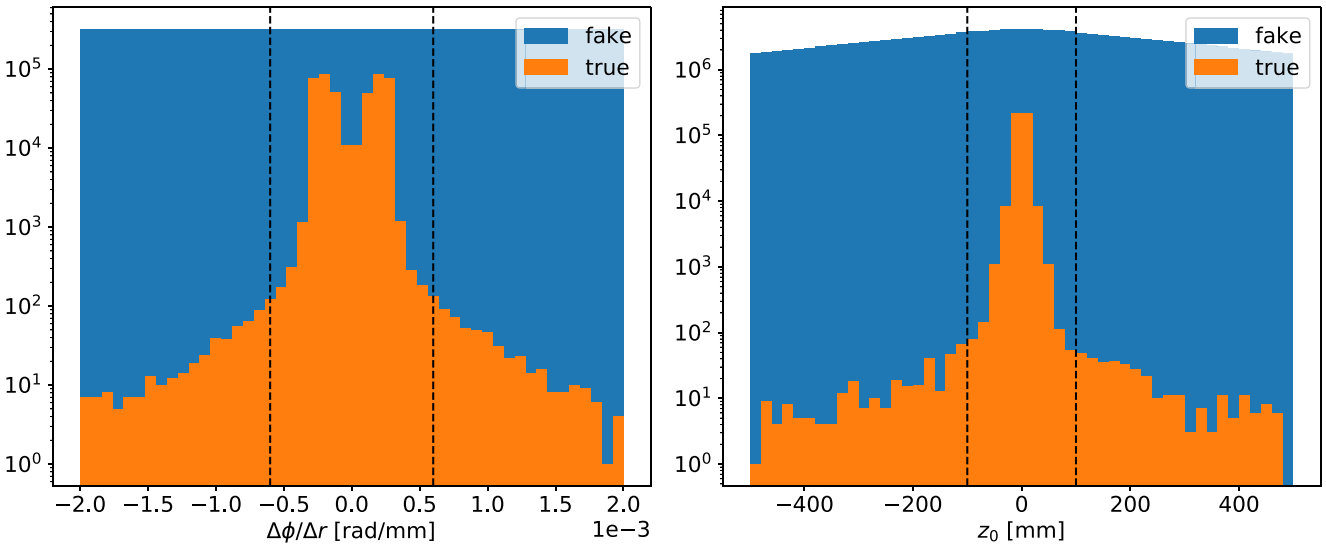


Fig. 12 Histogram of fake and true segments obtained during the graph construction procedure. The segments that reside inside the dashed lines are considered in order to maximize purity and efficiency. This step is important as it allows graph construction with fewer fake edges, and thus reduces computation times significantly

Appendix B: Expressibility and entanglement capability of PQCs

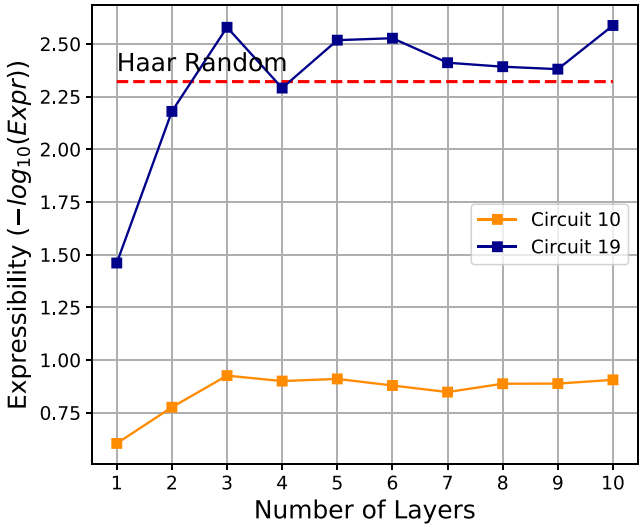


Fig. 13 Expressibility vs. number of layers for Circuit 10 (orange) and Circuit 19 (blue) in their 4 qubit configurations. Negative log of Expressibility is plotted for visual purposes

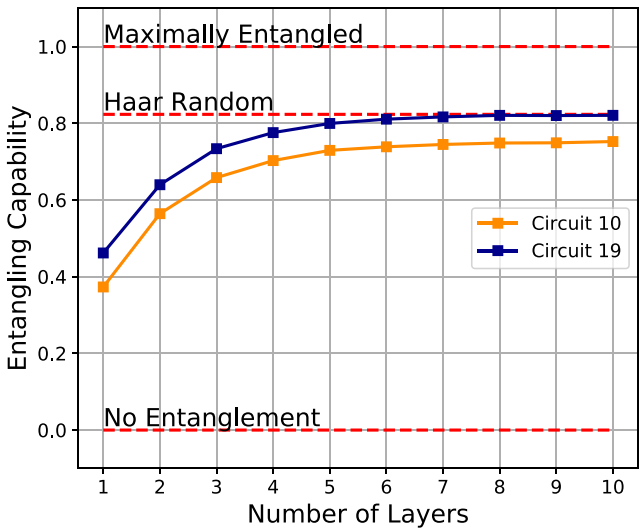


Fig. 14 Entanglement Capability vs. number of layers for Circuit 10 (orange) and Circuit 19 (blue) in their 4 qubit configurations

Appendix C: Hidden dimension size comparison

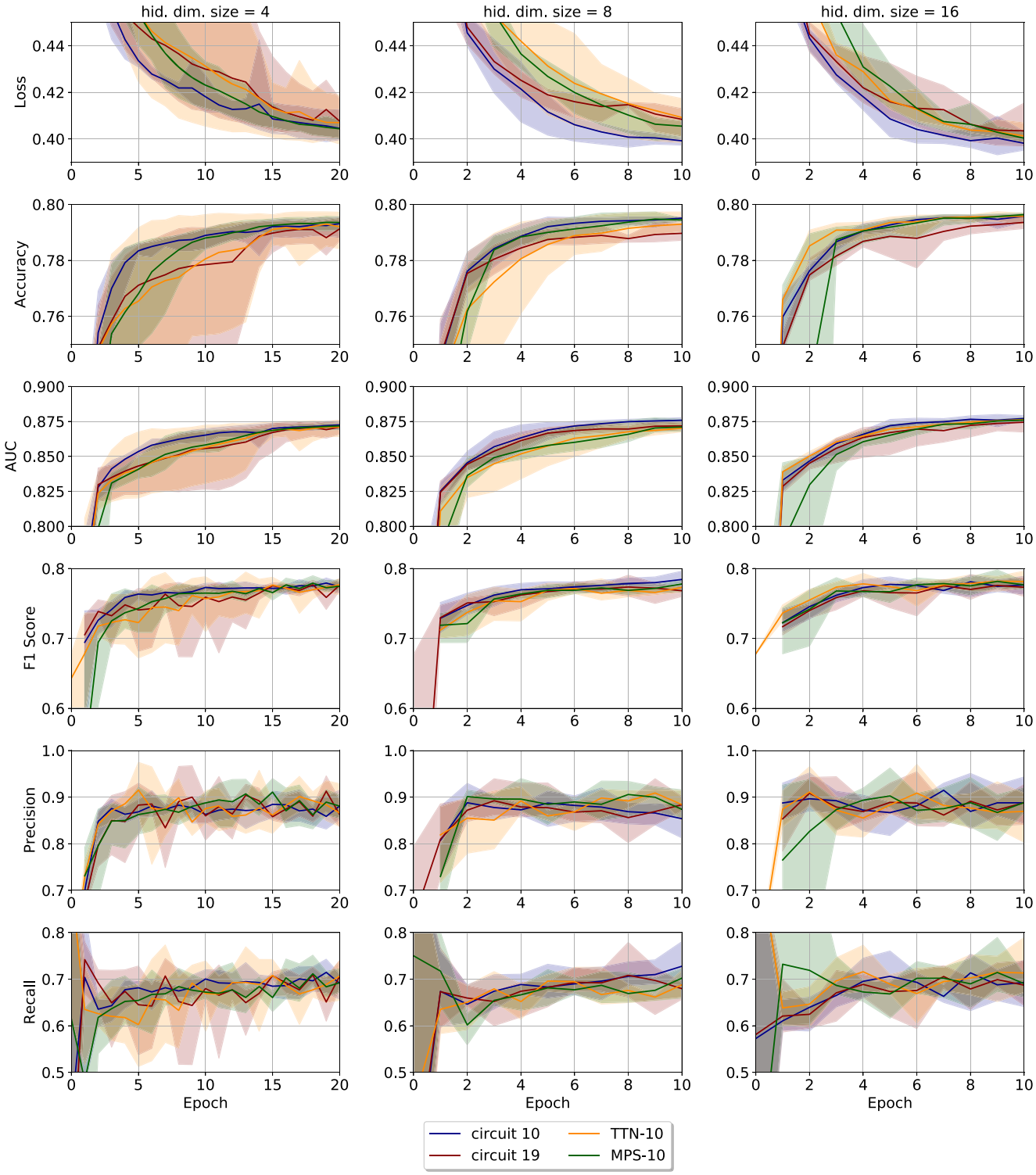


Fig. 15 Hidden dimension size comparison compares different hidden dimension sizes (N_D) by fixing $N_Q = N_D$, $N_I = 3$ and $N_L = 1$. The shaded regions represent the $\pm$ standard deviation of the best losses of all 5 runs. A 0.5 threshold is used for metrics that require a threshold

Appendix D: Number of layers comparison

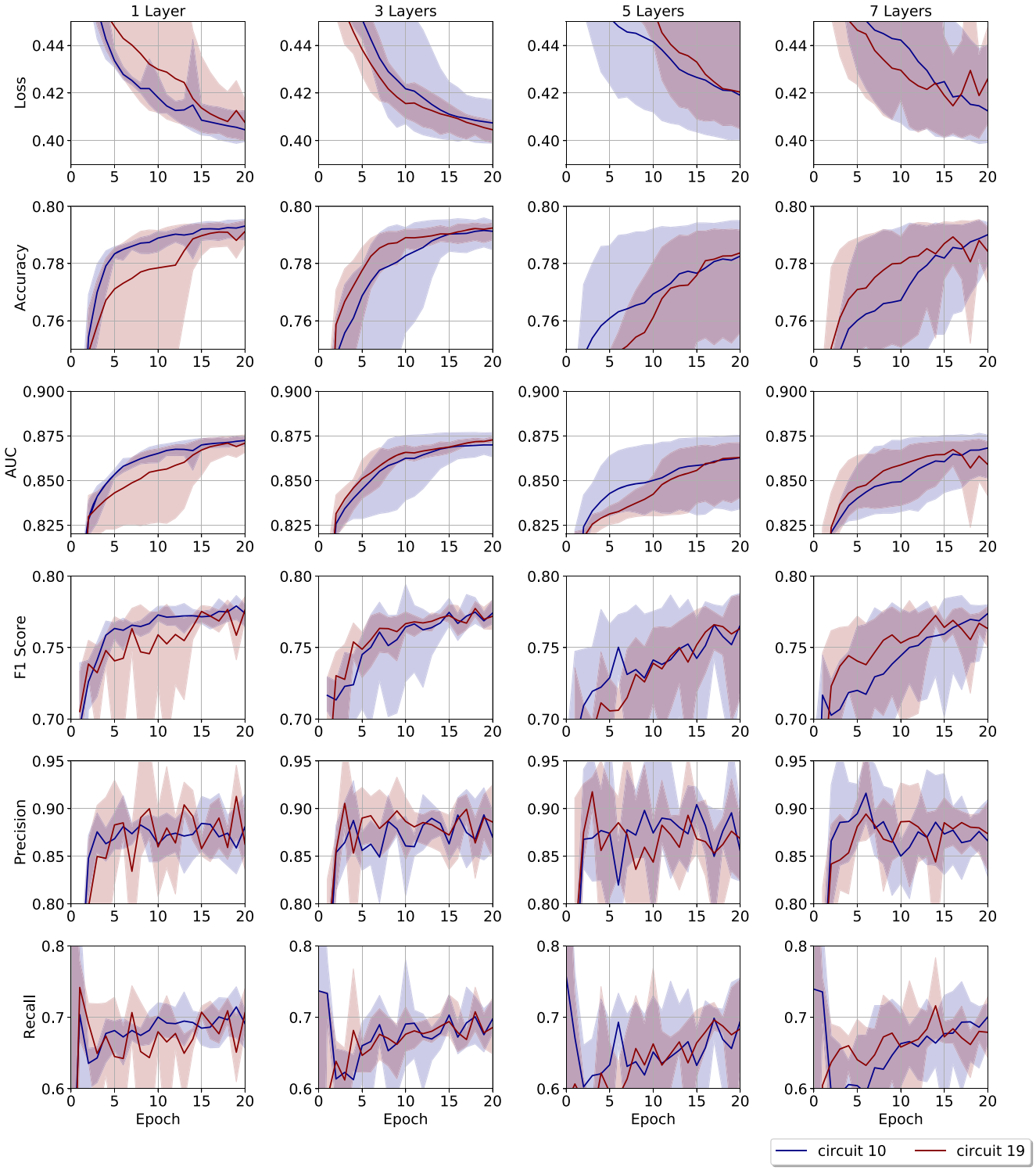


Fig. 16 Number of layers comparison compares number of layers by fixing $N_Q = N_D = 4$ and $N_I = 3$. The shaded regions represent the $\pm$ standard deviation of the best losses of all 5 runs. A 0.5 threshold is used for metrics that require a threshold

Appendix E: Number of iteration comparison

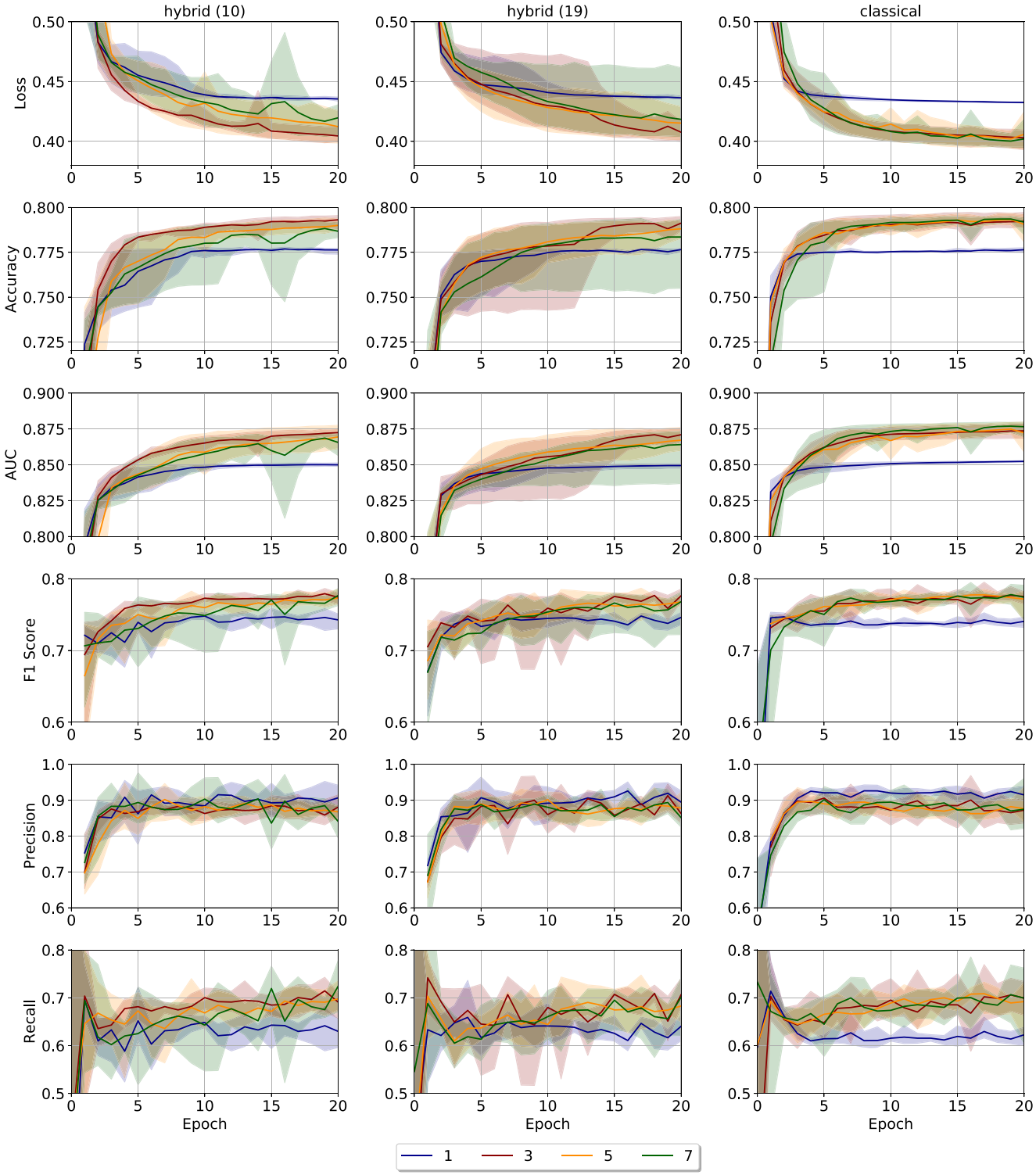


Fig. 17 Number of iteration comparison compares number of iteration by fixing $N_Q = N_D = 4$ and $N_L = 1$. The shaded regions represent the $\pm$ standard deviation of the best losses of all 5 runs. A 0.5 threshold is used for metrics that require a threshold

Appendix F: Embedding comparison

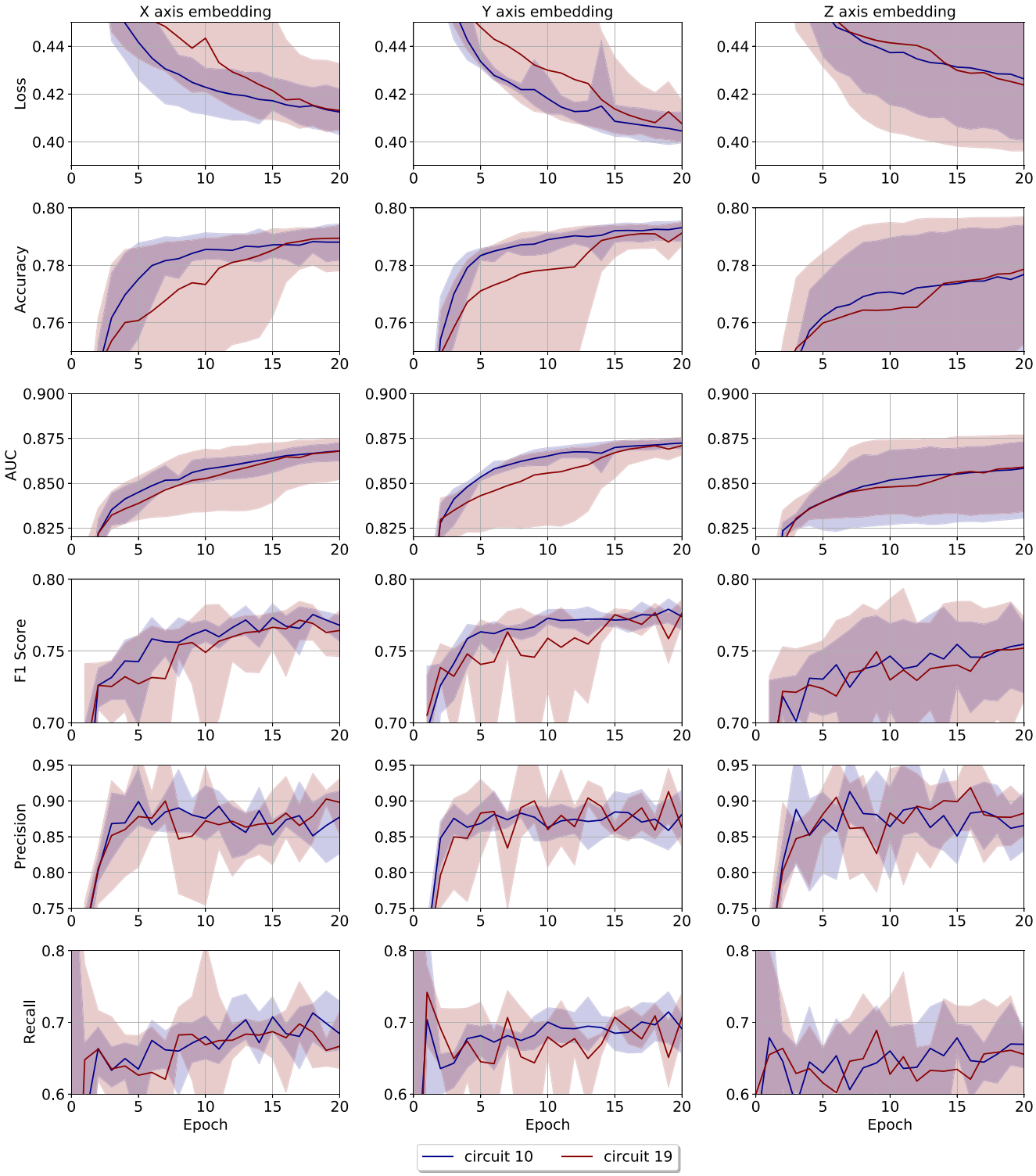


Fig. 18 Axis of angle embedding comparison compares different embedding axes by fixing $N_Q = N_D = 4$, $N_I = 3$ and $N_L = 1$. The shaded regions represent the $\pm$ standard deviation of the best losses of all 5 runs. A 0.5 threshold is used for metrics that require a threshold

Acknowledgements Authors would like to thank Alessandro Roggero for fruitful discussions. Part of this work was conducted at "*iBanks*", the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "*iBanks*".

Funding Open access funding provided by CERN (European Organization for Nuclear Research). This work was partially supported by the Turkish Atomic Energy Authority (TAEK) (Grant No: 2017TAEKCERN-A5.H6.F2.15 and 2020TAEK(CERN)-A5.H1.F5-26).

Software information The open-source software used in this work can be listed as follows. Python 3.8.5, NumPy v1.18.5 (Harris et al. 2020), Cirq v0.9.1 (Cirq Developers 2021), Tensorflow v2.3.1 (Abadi et al. 2016), Tensorflow Quantum v0.4.0 (Broughton et al. 2020), Scikit-learn v0.23.2 (Pedregosa et al. 2011), qpic v1.0.2 (Draper and Kutin 2020), Matplotlib v3.2.2 (Hunter 2007). The project codebase to reproduce all of the results presented here can be accessed through https://qtrkx.github.io.

Declarations

Conflict of interest The authors declare no competing interests.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp 265–283. arXiv:1605.08695

Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A, Woerner S (2021) The power of quantum neural networks. Nat Comput Sci 1(6): 403–409. https://doi.org/10.1038/s43588-021-00084-1

Albrecht J, Alves AA, Amadio G, Andronico G, Anh-Ky N, Aphecetche L, Apostolakis J, Asai M, Atzori L et al (2019) A roadmap for HEP software and computing R&D for the 2020s. Comput Softw Big Sci 3(1). https://doi.org/10.1007/s41781-018-0018-8

Amrouche S, Basara L, Calafiura P, Estrade V, Farrell S, Ferreira DR, Finnie L, Finnie N, Germain C, Gligorov VV et al (2019) The tracking machine learning challenge: accuracy phase. The Springer Series on Challenges in Machine Learning. https://doi.org/10.1007/978-3-030-29135-8_9. arXiv:1904.06778

Amrouche S, Basara L, Calafiura P, Emeliyanov D, Estrade V, Farrell S, Germain C, Vava Gligorov V, Golling T, Gorbunov S et al (2021) The tracking machine learning challenge : throughput phase. arXiv:2105.01160

Apollinari G, Brüning O, Nakamoto T, Rossi L (2015) High luminosity large hadron Collider HL-LHC. CERN Yellow Rep 5:1–19. https://doi.org/10.5170/CERN-2015-005.1, arXiv:1705.08830

Arrasmith A, Cincio L, Somma RD, Coles PJ (2020) Operator sampling for shot-frugal optimization in variational algorithms. arXiv:2004.06252

ATLAS Collaboration (2019) Fast track reconstruction for HL-LHC. Tech. Rep ATL-PHYS-PUB-2019-041, CERN, Geneva. https://cds.cern.ch/record/2693670

Bapst F, Bhimji W, Calafiura P, Gray H, Lavrijsen W, Linder L, Smith A (2019) A pattern recognition algorithm for quantum annealers. Comput Softw Big Sci 4(1):1. https://doi.org/10.1007/s41781-019-0032-5. arXiv:1902.08324

Benedetti M, Lloyd E, Sack S, Fiorentini M (2019) Parameterized quantum circuits as machine learning models. Q Sci Technol 4(4):043001. https://doi.org/10.1088/2058-9565/ab4eb5. arXiv:1906.07682

Bergholm V, Izaac J, Schuld M, Gogolin C, Blank C, McKiernan K, Killoran N (2018) PennyLane: automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968

Bhatia AS, Saggi MK, Kumar A, Jain S (2019) Matrix product state–based quantum classifier. Neural Comput 31(7):1499–1517. https://doi.org/10.1162/neco_a_01202, arXiv:1905.01426

Biscarat C, Caillou S, Rougier C, Stark J, Zahreddine J (2021) Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC. EPJ Web Conf 251:03047. https://doi.org/10.1051/epjconf/202125103047, arXiv:2103.00916

Bocci A, Innocente V, Kortelainen M, Pantaleo F, Rovere M (2020) Heterogeneous reconstruction of tracks and primary vertices with the CMS pixel tracker. Front Big Data 3:49. https://doi.org/10.3389/fdata.2020.601728, arXiv:2008.13461

Broughton M, Verdon G, McCourt T, Martinez AJ, Yoo JH, Isakov SV, Massey P, Niu MY, Halavati R, Peters E et al (2020) TensorFlow quantum: a software framework for quantum machine learning. arXiv:2003.02989

Campos E, Rabinovich D, Akshay V, Biamonte J (2021) Training saturation in layerwise quantum approximate optimization. Phys Rev A 104(3):L030401. https://doi.org/10.1103/PhysRevA.104.L030401, publisher: American Physical Society

Cao C, Wang X (2021) Noise-assisted quantum autoencoder. Phys Rev Appl 15(5):054012. https://doi.org/10.1103/PhysRevApplied.15.054012, publisher: American Physical Society

Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, McClean JR, Mitarai K, Yuan X, Cincio L, Coles PJ (2021) Variational quantum algorithms. Nat Rev Phys 3:625–644. https://doi.org/10.1038/s42254-021-00348-9

Chang SY, Herbert S, Vallecorsa S, Combarro EF, Duncan R (2021) Dual-parameterized quantum circuit GAN model in high energy physics. EPJ Web of Conf 251:03050. https://doi.org/10.1051/epjconf/202125103050. arXiv:2103.15470

Chen SYC, Wei TC, Zhang C, Yu H, Yoo S (2021) Hybrid quantum-classical graph convolutional network. arXiv:2101.06189

Cirq Developers (2021) Cirq. https://doi.org/10.5281/zenodo.4586899, See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors

Contardo D, Klute M, Mans J, Silvestris L, Butler J (2015) Technical proposal for the Phase-II upgrade of the CMS detector. Tech. Rep. CERN-LHCC-2015-010, LHCC-P-008, CMS-TDR-15-02, CERN. https://cds.cern.ch/record/2020886

Draper TG, Kutin SA (2020) <q—pic>: Quantum circuits made easy. https://github.com/qpic/qpic

Farhi E, Neven H (2018) Classification with quantum neural networks on near term processors. arXiv:1802.06002

Farrell S, Calafiura P, Mudigonda M, Prabhat, Anderson D, Vlimant JR, Zheng S, Bendavid J, Spiropulu M, Cerati G, Gray L, Kowalkowski J, Spentzouris P, Tsaris A (2018) Novel deep learning methods for track reconstruction. arXiv:1810.06111

Gentini L, Cuccoli A, Pirandola S, Verrucchi P, Banchi L (2020) Noise-resilient variational hybrid quantum-classical optimization. Phys Rev A 102(5):052414. https://doi.org/10.1103/PhysRevA.102.052414, publisher: American Physical Society

Grant E, Benedetti M, Cao S, Hallam A, Lockhart J, Stojevic V, Green AG, Severini S (2018) Hierarchical quantum classifiers. NPJ Quantum Inf 4(1):17–19. https://doi.org/10.1038/s41534-018-0116-9

Guan W, Perdue G, Pesah A, Schuld M, Terashi K, Vallecorsa S, Vlimant JR (2021) Quantum machine learning in high energy physics. Mach Learn Sci Technol 2(1):011003. https://doi.org/10.1088/2632-2153/abc17d. arXiv:2005.08582

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ et al (2020) Array programming with NumPy. Nature 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2

Heredge J, Hill C, Hollenberg L, Sevior M (2021) Quantum support vector machines for continuum suppression in B meson decays. arXiv:2103.12257

Hubregtsen T, Pichlmeier J, Stecher P, Bertels K (2021) Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. Quantum Mach Intell 3(1):9. https://doi.org/10.1007/s42484-021-00038-w

Hunter JD (2007) Matplotlib: A 2d graphics environment. Comput Sci Eng 9(3):90–95. https://doi.org/10.1109/MCSE.2007.55

Jang W, Terashi K, Saito M, Bauer CW, Nachman B, Iiyama Y, Kishimoto T, Okubo R, Sawada R, Tanaka J (2021) Quantum gate pattern recognition and circuit optimization for scientific applications. EPJ Web of Conf 251:03023. https://doi.org/10.1051/epjconf/202125103023. arXiv:2102.10008

Ju X, Farrell S, Calafiura P, Murnane D, Prabhat GL, Klijnsma T, Pedro K, Cerati G, Kowalkowski J et al (2020) Graph neural networks for particle reconstruction in high energy physics detectors. arXiv:2003.11603

Kingma DP, Ba J (2017) Adam: A method for stochastic optimization. arXiv:1412.6980

Larose R, Coyle B (2020) Robust data encodings for quantum classifiers. Phys Rev A 102(3):1–24. https://doi.org/10.1103/PhysRevA.102.032420. arXiv:2003.01695

Leymann F, Barzen J (2020) The bitter truth about gate-based quantum algorithms in the NISQ era. Quantum Sci Technol 5(4):044007. https://doi.org/10.1088/2058-9565/abae7d

Leyton-Ortega V, Perdomo-Ortiz A, Perdomo O (2021) Robust implementation of generative modeling with parametrized quantum circuits. Quantum Mach Intell 3(1):17. https://doi.org/10.1007/s42484-021-00040-2

Lucchesi D (2017) Computing resources scrutiny group report. Tech. Rep. CERN-RRB-2017-125 CERN, Geneva, Switzerland. http://cds.cern.ch/record/2284575

Magano D, Kumar A, Kālis M, Locāns A, Glos A, Pratapsi S, Quinta G, Dimitrijevs M, Rivošs A, Bargassa P, Seixas J, Ambainis A, Omar Y (2021) Investigating quantum speedup for track reconstruction: classical and quantum computational complexity analysis. arXiv:2104.11583

Mari A, Bromley TR, Killoran N (2021) Estimating the gradient and higher-order derivatives on quantum hardware. Phys Rev A 103(1):012405. https://doi.org/10.1103/PhysRevA.103.012405, publisher: American Physical Society

Marrero CO, Kieferová M, Wiebe N (2020) Entanglement induced barren plateaus. arXiv:2010.15968

McClean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. New J Phys 18(2):023023. https://doi.org/10.1088/1367-2630/18/2/023023

McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nat Commun 9(1):1–6. https://doi.org/10.1038/s41467-018-07090-4. arXiv:1803.11173

Mitarai K, Negoro M, Kitagawa M, Fujii K (2018) Quantum circuit learning. Phys Rev A 98(3):032309. https://doi.org/10.1103/PhysRevA.98.032309

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems, vol 32. Curran Associates Inc., pp 8024–8035

Pata J, Duarte J, Vlimant JR, Pierini M, Spiropulu M (2021) MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. Eur Phys J C 81(5):381. https://doi.org/10.1140/epjc/s10052-021-09158-w. arXiv:2101.08578

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Pesah A, Cerezo M, Wang S, Volkoff T, Sornborger AT, Coles PJ (2020) Absence of barren Plateaus in quantum convolutional neural networks. arXiv:2011.02966

Romero J, Aspuru-Guzik A (2021) Variational quantum generators: generative adversarial quantum machine learning for continuous distributions. Adv Quantum Technol 4(1):2000003. https://doi.org/10.1002/qute.202000003

Schuld M, Sweke R, Meyer JJ (2021) Effect of data encoding on the expressive power of variational quantum machine learning models. Phys Rev A 103(3):032430. https://doi.org/10.1103/PhysRevA.103.032430

Shapoval I, Calafiura P (2019) Quantum associative memory in hep track pattern recognition. EPJ Web of Conf 214:01012. https://doi.org/10.1051/epjconf/201921401012. arXiv:1902.00498

Sharma K, Khatri S, Cerezo M, Coles PJ (2020) Noise resilience of variational quantum compiling. New J Phys 22(4):043006. https://doi.org/10.1088/1367-2630/ab784c, publisher: IOP Publishing

Shlomi J, Battaglia P, Vlimant JR (2021) Graph neural networks in particle physics. Mach Learn Sci Technol 2(2):021001. https://doi.org/10.1088/2632-2153/abbf9a, arXiv:2007.13681

Sim S, Johnson PD, Aspuru-Guzik A (2019) Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Adv Quantum Technol 2(12):1900070. https://doi.org/10.1002/qute.201900070, arXiv:1905.10876

Suzuki Y, Kawase Y, Masumura Y, Hiraga Y, Nakadai M, Chen J, Nakanishi KM, Mitarai K, Imai R, Tamiya S et al (2020) Qulacs: a fast and versatile quantum circuit simulator for research purpose. arXiv:2011.13524

Terashi K, Kaneda M, Kishimoto T, Saito M, Sawada R, Tanaka J (2021) Event classification with quantum machine learning in high-energy physics. Comput Softw Big Sci 5(1):2. https://doi.org/10.1007/s41781-020-00047-7, arXiv:2002.09935

The ATLAS Collaboration (2015) ATLAS Phase-II upgrade scoping document. Technical Report CERN-LHCC-2015-020.LHCC-G-166 CERN, Geneva, Switzerland. https://cds.cern.ch/record/2055248

Tüysüz C, Carminati F, Demirköz B, Dobos D, Fracas F, Novotny K, Potamianos K, Vallecorsa S, Vlimant JR (2020a) Particle track reconstruction with quantum algorithms. EPJ Web Conf 245:09013. https://doi.org/10.1051/epjconf/202024509013

Tüysüz C, Carminati F, Demirköz B, Dobos D, Fracas F, Novotny K, Potamianos K, Vallecorsa S, Vlimant JR (2020b) CTD2020: A quantum graph network approach to particle track reconstruction. https://doi.org/10.5281/zenodo.4088474, arXiv:2007.06868

Tüysüz C, Novotny K, Rieger C, Carminati F, Demirköz B, Dobos D, Fracas F, Potamianos K, Vallecorsa S, Vlimant JR (2020c) Performance of particle tracking using a quantum graph neural network. arXiv:2012.01379

Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. arXiv:1710.10903

Verdon G, McCourt T, Luzhnica E, Singh V, Leichenauer S, Hidary J (2019) Quantum graph neural networks. arXiv:1909.12264

Wang S, Fontana E, Cerezo M, Sharma K, Sone A, Cincio L, Coles PJ (2021) Noise-induced barren plateaus in variational quantum algorithms. arXiv:2007.14384

Wu SL, Chan J, Guan W, Sun S, Wang A, Zhou C, Livny M, Carminati F, Di MeglioA, Li ACY, Lykken JD, Spentzouris P, Chen SYC, Yoo S, Wei TC (2021a) Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. J Phys G: Nuclear Part Phys. https://doi.org/10.1088/1361-6471/ac1391, arXiv:2012.11560

Wu SL, Sun S, Guan W, Zhou C, Chan J, Cheng CL, Pham T, Qian Y, Wang AZ, Zhang R, Livny M, Glick J, Barkoutsos PK, Woerner S, Tavernelli I, Carminati F, Di MeglioA, Li ACY, Lykken J, Spentzouris P, Chen SYC, Yoo S, Wei TC (2021b) Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC. Phys Rev Res 3(3):033221. https://doi.org/10.1103/PhysRevResearch.3.033221, arXiv:2104.05059

Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021c) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Zhang K, Hsieh MH, Liu L, Tao D (2020) Toward trainability of quantum neural networks. arXiv:2011.06258

Zhao C, Gao XS (2021) Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus. Quantum 5:466. https://doi.org/10.22331/q-2021-06-04-466

Zlokapa A, Anand A, Vlimant JR, Duarte JM, Job J, Lidar D, Spiropulu M (2019) Charged particle tracking with quantum annealing-inspired optimization. arXiv:1908.04475