# Exploring hybrid quantum-classical neural networks for particle tracking

## SEPTEMBER 2020

**AUTHOR:**
Carla Rieger

ETH Zürich, Switzerland

**SUPERVISORS:**
Sofia Vallecorsa,
Daniel Dobos,
Kristiane Novotny

# ABSTRACT

The High Luminosity Large Hadron Collider (HL-LHC) at CERN will involve a significant increase in the complexity and sheer size of data with respect to the current LHC experimental complex. Hence, the task of reconstructing the particle trajectories will become more and more complex due to the number of simultaneous collisions and the resulting increased detector occupancy. Aiming to identify the particle paths, machine learning techniques such as graph neural networks are being explored in the HEP.TrkX project and its successor, the Exa.TrkX project. Both show promising results and reduce the combinatorial nature of the problem. Previous results of our team have demonstrated the successful attempt of including quantum computing concepts within graph neural networks that are able to reconstruct the particle track based on the hits of the detector. A higher overall accuracy is gained by representing the training data in a meaningful way within an embedded space. That has been included in the Exa.TrkX project by applying a classical MLP. Consequently, pairs of hits belonging to different trajectories are pushed apart while those belonging to the same ones stay close together. We explore the applicability of quantum circuits within the task of embedding using hybrid-classical neural network architectures and show preliminary results.

# TABLE OF CONTENTS

## 1. INTRODUCTION

With the start of the High Luminosity LHC (HL-LHC), there will be much more simultaneous collisions (pile-ups) leading to ambiguities making the task of reconstructing the particle tracks highly complex [1]. To explore how (classical) machine learning techniques can tackle this task, the TrackML challenge has been launched on Kaggle [2]. This dataset consists of more than 8k simulated collision events. Since its release, the dataset has become important an important benchmark for particle tracking algorithms.

There are several classical machine learning approaches, such as graph neural networks that show a great performance in identifying the trajectories from the simulated detector measurements. They are investigated by the HEP.TrkX project [3] and its update, the Exa.TrkX project [4]. In both of these projects, the graph neural network has a general structure of iteratively applying a node and an edge network. The edge and node information can either take various numbers of nodes into account. Mostly, doublets or triplets are taken into account when constructing a graph. Doublets are built up of 2 hits (i.e., a single detector measurement within a layer) that correspond to nodes and are connected by an edge. The edge can be either part of a real trajectory or not. The same principle holds for triplets, but the focus is on 3 hits corresponding to 3 nodes connected with 2 edges that can either be part of a true particle trajectory or not.

The Exa.Trkx project uses a more advanced preprocessing compared to the HEP.TrkX project. An important part in the new data processing pipeline forms the embedding of the detector measurement data. A feed-forward neural network with hidden layers – referred to as multilayer perceptron or MLP – acts as a non-linear projection onto a higher-dimensional embedding space. Hits belonging to the same trajectory are embedded close together, while those belonging to different trajectories are embedded far apart. This step improves the performance of upcoming steps, especially in the classification task.

Building up on the HEP.TrkX project, quantum graph neural networks, that use quantum circuits instead of classical neural networks to process the edge and node information, have shown promising preliminary results in identifying the particle paths [5], [6], [7]. Hence, it is a really interesting part of research to examine the behavior of quantum circuits within applications using neural networks.

This work explores hybrid quantum-classical networks for embedding the simulated detector measurements of the TrackML dataset, building up on the projects previously mentioned. Several quantum circuit configurations have been explored that extend or replace parts of the classical MLP used within the Exa.TrkX project. This quantum-classical version utilizes the exponential size of the Hilbert space and explores the effects of entanglement on the embedding and inference tasks across different sized classical networks and quantum circuits.

This work is structured as follows. First, we introduce the idea of gated quantum circuits which is used in quantum machine learning models (Section 2) and the specification of the TrackML dataset (Section 3) used for training the different models. In Section 4, we focus on the various implemented quantum circuit architectures within the (quantum) neural networks. For each of these architectures, the training results are presented in Section 5. We conclude this work with an outlook in the Discussion part (Section 6).

## 2. QUANTUM GATES AND CIRCUITS

This section provides a short overview on how quantum computing can be included within classical neural networks. Precisely, the exact use of gate-based quantum circuits that extend a classical neural network to a hybrid quantum-classical network.

Quantum circuits represent a scheme on how to act on an initially specified number of qubits. They are executed from left to right. Each qubit is in a state that can be represented on the Bloch sphere. Initially all qubits in a quantum circuit are initialized in the $|0\rangle$ state, point upwards in the Bloch representation. Then single-qubit gates such as rotational gates or 2-qubit gates that can entangle different qubits, e.g. the CNOT gate can be applied. In this implementation, those rotational gates are used to apply a certain rotation by a specified angle and a certain direction. Rotation operators around the $\hat{x}$-axis act on the $|0\rangle$ state as follows [8] :

$$\mathrm{R}_x(\theta)|0\rangle = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ -i\sin(\theta/2) \end{pmatrix}$$

Here, $X$ represents the respective Pauli matrix. The gates applied on the different qubits in a quantum circuit are designed in a reversible way, meaning the output of the gate fully determines its input. There is no loss of information in a noiseless quantum circuit. The 2-qubit gates used within this work are conditional gates that entangle two qubits within the circuit. The CNOT gate (conditional not operation) can be written in matrix form as shown here:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

By encoding information coming from the previous classical layers, the circuit may act as a function. Moreover, trainable parameters can be included within the quantum circuit. In this case, gates included in the circuit that exhibit a variable parameter are initialized at random and are optimized during the training procedure, similarly to the classical weights in a neural network. Moreover, a quantum circuit can include hidden dimensions consisting of additional qubits that do not encode input information, i.e., by using additional ancilla qubits. They can be entangled (i.e. exhibit strong correlations) with other “non-hidden or hidden qubits”. Gates with trainable parameters may be applied on the hidden dimensions to expand the number of free parameters in the circuit.

## 3. THE DATASET

The network has been trained on the TrackML dataset that is publicly available [9]. This data consists of more than 8k simulated events (collisions of proton bunches) of detector measurements (hits). The task is to classify the 3-dimensional hits in order to identify the trajectories of the single particles involved in the collision.

The figure below (Fig. 1) displays how the hits, displayed as dots, are connected by edges and form trajectories. True trajectories are displayed in blue and false ones in red.

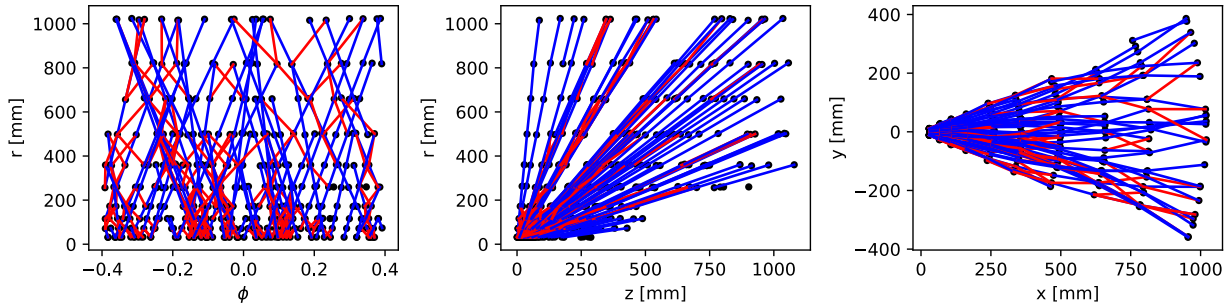



*Figure 1. Illustration of the particle track reconstruction task. The black dots represent the simulated detector hits connected to trajectories where true edges are displayed in blue and false edges in red [6].*

Due to training time restrictions only 20% of the TrackML dataset have been processed to doublets in the same way as the Exa.TrkX project. 10k doublets have been generated that are used for training the hybrid

quantum-classical network as well as the classical version for comparison (8k for training, 2k for validation). This is a relatively small number of chosen samples to have a reasonable training time for testing purposes. In further tests, more samples should be used for training. For the train-validation data split, the data has been shuffled and thus randomly assigned to one of the datasets to guarantee an expressive validation procedure. By observing the behaviour of training and validation loss, overfitting can be seen and prevented. Thus, the validation loss is more informative than the actual training loss because the performance is evaluated on a separate, independent part of the data that is not involved in the optimization procedure. Hence, only the validation loss is shown in the results.

## 4. HYBRID ARCHITECTURE

The architecture follows two different approaches that extend the classical multilayer perceptron (MLP) used for embedding by the Exa.TrkX project [4]. Both approaches follow the general structure as presented in Fig. 2 with different quantum circuits (QC) and various numbers of hidden layers ($n_{layers}$) of the MLP.

In the first approach, the classical MLP is combined with different quantum circuits acting as a function, where the output from the previous classical part is encoded using rotational gates, as described above. Hence, the output layer of the classical MLP projects onto the number of trainable parameters $n_{parameters}$ of the quantum circuits. Depending on the circuit that is used, conditional rotational gates or CNOT gates introduce a certain degree of entanglement between the qubits. Quantum circuits that exhibit different behaviors with respect to entanglement, expressibility and varying numbers of parameters [10] have been used to explore how the circuit-dependent level of entanglement and expressibility influences the training behavior of the MLP.

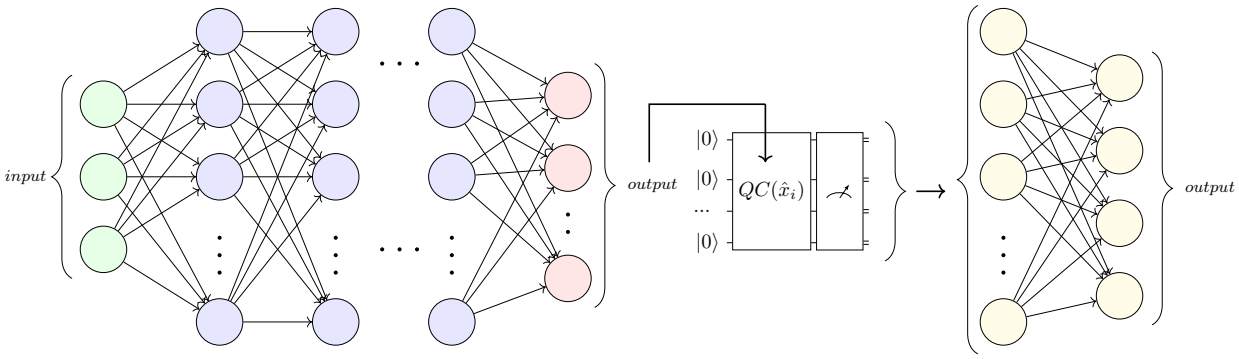



*Figure 2. Illustration of the general structure of the used hybrid quantum-classical MLP. The displayed quantum circuit (QC) acts as a placeholder for the quantum circuit architectures presented below.*

The second approach using the quantum feature map follows again the same general structure as described in Fig. 2. The output of the classical MLP is encoded using rotational gates. Furthermore, this version includes trainable parameters that are optimized during the training procedure. As before, the last projection, depicted in yellow, projects the output of the quantum circuit ($n_{measurements}$) onto the preferred embedding dimension that is smaller or equal to $n_{measurements}$.

## a. The classical MLP

The classical MLP used in the Exa.TrkX project [4] has hidden layers of the dimension of $n_{layers} \times 512$, where 512 is the number of neurons per layer and $n_{layers} = 10$. There are additional input and output layers. The input layer projects the 3-dimensional input data onto the size of the first hidden layer and the output layer projects onto the output dimension. The Exa.TrkX project uses an embedding or output dimension of *8*. The classical embedding version performs really well and forms a successful pre-processing step, that leads to advantages in later steps. The aim is to explore how additional quantum circuits change the behaviour and how the replacement of classical layers with quantum circuits performs in simulations.

## b. Quantum circuit approach

The general architecture can be seen in Fig. 2. For this first approach $n_{layers}$ is set to 10 and the quantum circuits from Fig. 3 act as an encoding function. The 3-dimensional input data from the TrackML dataset is hereby embedded into a 4-dimensional space. Due to the long training time for the simulation of the quantum circuit, in the beginning only 4 and 8 qubit quantum circuits have been used. This limits the embedding space dimension for this 4-qubit circuit to $D_{out} = 4$. The output dimension $D_{out}$ was kept constant also for the 8 qubit circuits for better comparison and additionally influence of the down-projections was investigated (from 8 measurements to $D_{out} = 4$). The circuits should be later extended to ones with more than 8 qubits and give an embedding into a higher dimensional space comparable to the 8-dimensional space used in the Exa.TrkX project.

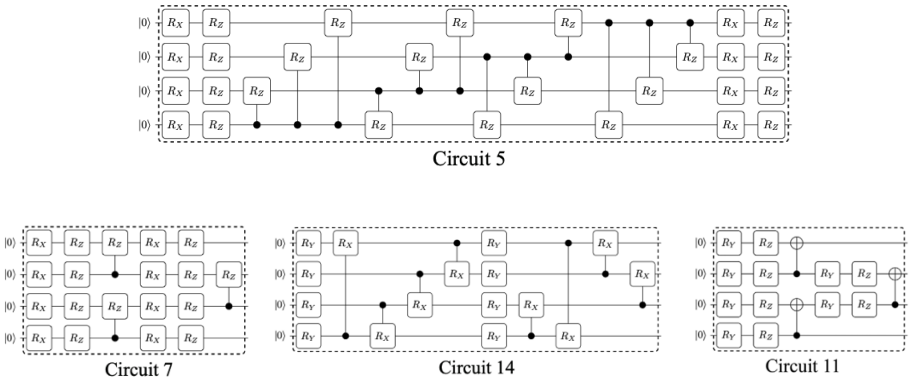



*Figure 3. The 4-qubit quantum circuits [10] that have been used within the general architecture (Fig.2).*

The different quantum circuits displayed in Fig. 3 have been used as QC in the hybrid MLP architecture displayed in Fig. 2. Those three circuits have been chosen due to their differences with respect to entanglement and expressibility as depicted in [10]. Entanglement is measured using the Meyer-Wallach entanglement measure and expressibility as the Kullback-Leibler-divergence (KL-divergence) between the estimated probability distribution of the fidelities of the respective quantum circuit and the distribution of Haar random states. The rotational gates (RX and RY gates) and conditional rotations exhibit variable parameters (angle of the rotation). In this approach, they are used as an encoding function meaning that the output of the previous classical MLP layers acts as an input to those parameter-gates. Thus, the quantum circuit acts as a function:

$$\mathrm{QC} : \mathrm{R}^{n_{parameters}} \rightarrow \mathbb{R}^{n_{measurements}}, \mathrm{QC} \in \{5, 7, 11, 14\}$$

and projects the input down to the number of measurements in the circuit. There is a further down-projection into a lower-dimensional embedding space applied if the number of measurements is higher when using more qubits (i.e. using the 8-qubit circuit and an embedding dimension of 4).

## c. Quantum feature map approach

The second approach uses a quantum circuit that includes trainable parameters. The initial architecture of the quantum circuit was adapted from [11]. The overall structure of the hybrid network follows again Fig. 2. In this version, the output of the classical MLP is encoded repeatedly using rotational gates. In between, entangling gates are applied that include parameters that can be optimized. Various numbers of hidden dimensions can be included. To keep the size of the circuit as small as possible, only one hidden dimension was used for testing purposes here. The hidden dimension here is a single qubit that is not used for encoding or decoding but is entangled with other non-hidden dimensions and includes trainable parameters in form of variable angles of rotational gates. Furthermore, this approach can be extended to more qubits and hence encode more input parameters as well as include more trainable parameters, i.e. within the hidden dimensions.

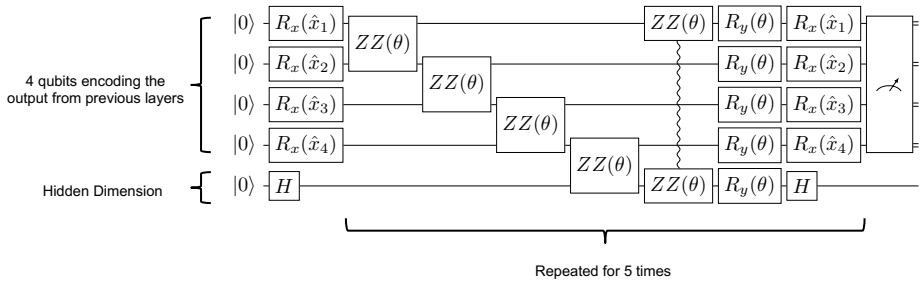



*Figure 4. Quantum circuit of the quantum feature map approach used within the general architecture in Fig. 2. Adapted from [11].*

The quantum circuit of the quantum feature map approach is displayed in Fig. 4 and exhibits 75 parameters. The RX gates encode the input coming from 4-dimensonal previous layers and the remaining parameters are variable and hence randomly initialized and trainable. Those values are optimized during training. Just like before, each qubit is initialized in the $|0\rangle$ state in the beginning. The first 4 qubits are the ones that are measured at the end of the quantum circuit, the 5$^{th}$ one forms the hidden dimension of the circuit. It is entangled with the other qubits and exhibits variable parameters.

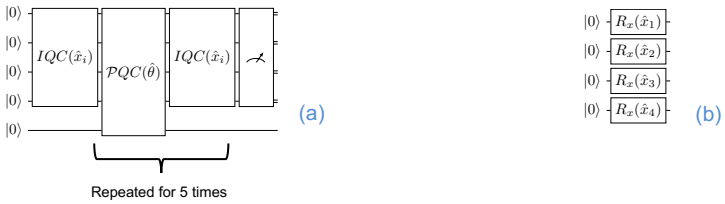



*Figure 5. (a) Simple schematics of the circuit displayed above. IQC is the quantum circuit encoding the input parameters and PQC the parametric quantum circuit that exhibits trainable parameters. (b) Encoding structure using rotational gates that builds up the IQC.*

Fig. 5a presents the complex quantum circuit from Fig. 4 in a simplified way. It is visible how the input (IQC) is repeatedly applied in every iteration. This encoding is done via rotational gates as shown in Fig. 5b.

## 5. RESULTS AND IMPLEMENTATION DETAILS

The different architectures for the hybrid quantum-classical neural network described above have been tested and the results are shown in detail below. The simulations of the hybrid quantum-classical network structure have been done in Python using PyTorch [12] and quantum computing libraries such as Pennylane [13] and Qiskit [14]. This is a supervised training procedure, i.e. there is an output given and with respect to that the loss function is evaluated and the parameters changed in a way such that the loss decreases.

The dataset consists of doublets, which are two 3-dimensional points (hits) in a Euclidean space that form nodes connected with an edge. Each of these doublets have a label that defines whether they are true edges, i.e., belong to the same trajectory, or not. The aim of the non-linear transformation that is applied on this doublet input data aims to represent those doublets, i.e. the hits in a feature space where the true doublets are close together and the false ones are further apart. This is done by using the hinge embedding loss that was also used in the Exa.TrkX project. For the *n*-th sample consisting of the two points in 3-dimensional space forming a doublet and their label using the hinge embedding loss, available in PyTorch [12], this is:

$$sample_n = [(x_i, x_j), y_{i,j}]$$

$$loss(sample_n) = \begin{cases} \max\{0, 1 - \|\Phi(x_i, \theta) - \Phi(x_j, \theta)\|_2\}, & \text{if } (x_i, x_j) \text{ belong to the same trajectory} \\ 1, & \text{if } (x_i, x_j) \text{ do not belong to the same trajectory} \end{cases}$$

The *n*-th sample is described by two points in the original space, denoted as $(x_i, x_j)$. To each of these doublet point pairs belongs a label $y_{i,j} \in \{-1, 1\}$ that indicates if the two points belong to the same trajectory $y_{i,j} = 1$ or not $y_{i,j} = -1$. The hinge embedding loss function favors in case where two points belonging to the same trajectory are embedded close together. Other loss functions, like the cosine embedding loss have been tested.

### a. Training the quantum circuit construction

The quantum circuit architecture (displayed in Fig. 2 and 3) was trained using Adamax optimizer [15], a learning rate of 0.001, batch size of N = 100. The classical MLP has $n_{layers} = 10$ where each of the layers consist of $512$ neurons.

| *Circuit number* | *Number of parameters* $n_{parameters}$ | *Number of gates (1+2 qubit gates)* | *Entanglement (the higher the better) [16]* | *Expressibility (the lower the better) [16]* | *Average training time per batch* |
|---|---|---|---|---|---|
| *5* | *28* | *28* | *0.29* | *0.05* | *37 s ± 8 s* |
| *7* | *19* | *19* | *0.21* | *0.10* | *20 s ± 4 s* |
| *11* | *12* | *15* | *0.54* | *0.13* | *14 s ± 4 s* |
| *14* | *16* | *16* | *0.49* | *0.02* | *16 ± 4s* |

*Figure 6. Comparison between the different 4-qubit circuit parameters. Circuits from [10].*

In Fig. 6, a comparison between the quantum circuits that have been trained is displayed. The training time increases with increasing number of gates in the 4-qubit circuits. Computationally expensive are also the quantum gates applied on 2 qubits. The difference in number of gates and number of parameters occurs because some of the gates (e.g. the CNOT gate) do not have a variable parameter. The listed quantum circuits have been chosen due to their different values regarding the metrics of entanglement and expressiblity. The presented values have been reproduced and the metrics are described in more detail in [10]. As a higher value with respect to the entanglement metric and a lower expressibility value are preferred, we expect a good performance of circuit 14. Indeed, this is the case as seen from the validation loss displayed in the figure below.

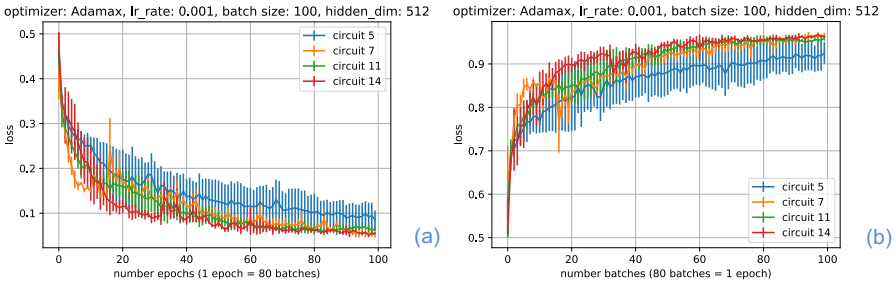



*Figure 7. Validation loss (a) and score (b) of the hybrid network with the respective quantum circuit within the general architecture. The training parameters are displayed on top of the figure.*

In Fig. 7, the mean values of 3 independent runs with different random states and the respective standard deviation as errorbars are shown. We observe a strong initialization dependence for circuit 5 that also exhibits the most parameters of the tested circuits, in detail discussed below. Circuit 14 performs the best converging to the lowest validation loss. While circuit 7 is converging to a similarly low value, the validation loss function exhibits higher and lower spikes. This could be explained by the training strategy using mini-batches of size 100. Circuit 11 converges to a comparable value of validation loss but seems to be converging more slowly until epoch 40. An unfortunate random initialization might explain the initially slow convergence rate of circuit 7 and especially 5. The corresponding score was computed as for [4]. The training results can be improved, by repeatedly applying the respective quantum circuit [10]. By doing this, the entanglement value increases as the KL-divergence decreases, i.e., the expressibility of the circuit increases. The expressibility is defined as the KL-divergence between the uniform distribution of states (ensemble of Haar random states) and the estimated fidelity distribution of the quantum circuit [10]. Hence, the flexibility of the circuits increases as the KL-divergence decreases and the expressibility rises as the quantum circuits are able to represent more states. This improvement over the number of repetitions of the circuit can be seen in Fig. 8.

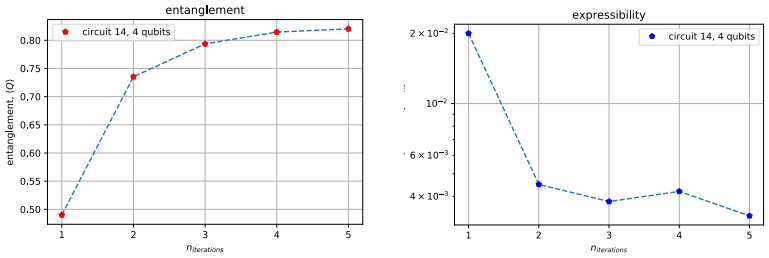



*Figure 8. Change in entanglement (a) and expressibility (b) value for quantum circuit (QC) 14 which is repeated for $n_{iterations}$. Reproduced values from [4].*

Hence, repeating the best performing circuit 14 from before can improve the validation loss and score displayed above. But such a repetition increases the number of gates in the circuit which leads to longer training times.

### i. Barren plateaus

Even though the hybrid quantum-classical neural network version uses shallow quantum circuits including only up to 8 qubits and the number of parameters is of $O(10)$, the training performance is highly dependent on the initialization of the circuit parameters. In some cases, the training and validation loss does not converge at all within the 100 epochs that have been used for training.

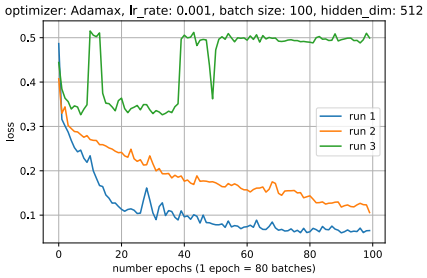
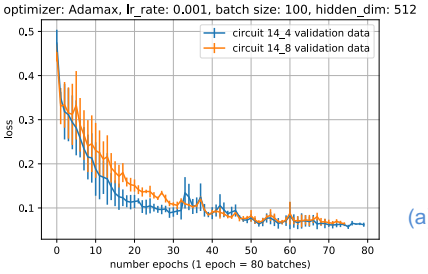



*Figure 9. Validation loss of 3 independent runs of hybrid network using circuit 5 with 4 qubits.*

It is visible in Fig. 9 how much the training success depends on the initialization of the parameters, i.e. convergence of the loss function within the first 100 epochs. The loss function shows the behaviour of 3 independent runs with different random state for initialization of the parameters in the classical MLP. Those random parameters act at the same time as the input to the quantum circuit and thus lead in case of *run 3* to an unfavourable convergence behaviour, while for *run 2* the initialization seems to only slow down the convergence in comparison to *run 1*. Several approaches exist to avoid such barren plateaus [17]. The spikes could be explained by the mini-batch optimization procedure when using Adamax optimizer. *Run 3* was omitted in Fig. 7 for better visualization.

### ii. Expanding the number of qubits

The initial circuits are all constructed using 4 qubits corresponding to 4 qubits in the quantum circuit. If all qubits are measured, the output is also 4-dimensional, and the input dataset is embedded into a 4-dimensional space. Thus, enlarging the quantum circuit allows to embed into higher dimensional spaces, since the Exa.TrkX version embeds into an 8-dimensional space. To compare the different performance in validation loss and running time, for the version including the 8-qubit quantum circuit, the last layer projects the output of the quantum circuit ($n_{measurements} = 8$) onto the same 4-dimensional embedding space.




(a)

| $QC$ | $n_{parameters}$ | $t_{batch}(in\ min.)$ |
|---|---|---|
| *Circuit 14, 4 qubits* | 16 | $0.28 \pm 0.07$ |
| *Circuit 14, 8 qubits* | 32 | $1.44 \pm 0.49$ |

(b)

*Figure 10. Validation loss of the hybrid architecture including circuit 14 using 4 and 8 qubits (a) and the respective running times per batch (b). The training parameters are displayed on top of figure (a).*

Fig. 10 shows the validation loss of the two versions of circuit 14 with 4 and 8 qubits. They exhibit a similar training performance while the 8-qubit circuit converges slightly slower than the 4 qubit one. The plot shows the mean value of 3 independent runs of one batch per epoch and the error bars display the respective standard deviation regarding the 3 runs. Due to the difference in number of qubits and number of parameters, the training time per batch for the 8-qubit circuit is much longer. In this case, the performance difference doesn't justify increasing the qubit number when projecting onto a 4-dimensional embedding space due to the difference in simulation times. But the 8-qubit version becomes important because of the higher number of measurements. With this circuit, it is possible to embed onto an 8-dimensional space in comparison to only 4 dimensions in the 4-qubit version.

## b. Training the quantum feature map approach

The quantum feature map network was trained using the same specifications as before. Again, Adamax was used as optimizer, a learning rate of 0.001 and the batch size of N = 100. The embedding dimension here is also 4, limited due to the number of measurements within the quantum circuit and training time constraints.

### i. Comparison of different sizes of the classical MLP

As the quantum feature map exhibits trainable parameters, it is an interesting question, how the performance changes when the network replaces classical layers of the MLP. Hence, a small range of combinations of layers in comparison to the classic 10-layer case have been tested. The parameter and values with respect to the entanglement and expressibility metrics are shown in Fig. 12b). Due to the long training times, this approach was only trained until epoch 25, hence adding more epochs could change the results drastically.

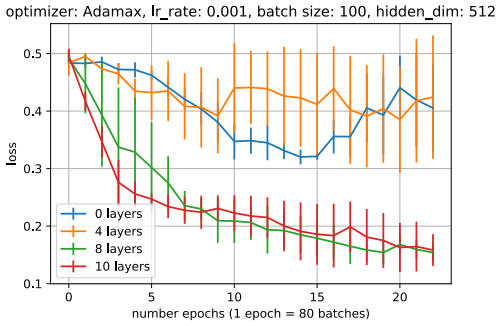
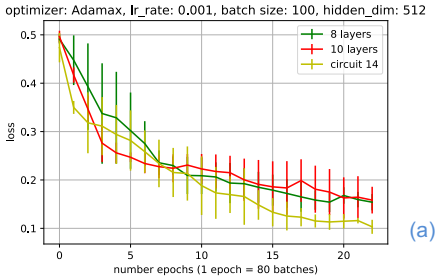



*Figure 11. Validation loss and training times of the quantum feature map approach with variable number of classical layers within the MLP. The training parameters are displayed on top of the figure.*

As displayed in Fig. 11, the 8-layer version seems to perform better or at least is comparable to the 10-layer version in these early epochs. The 4-layer version seems to converge to a very high validation loss value and thus could reach its expressiveness quite early. The 0-layer variant proves that there is an optimization over the variables in the quantum circuit and hence a non-zero gradient. The expressiveness in this case is also limited due to the low number of parameters in the classical and quantum circuit. Moreover, both variants (0- and 4-layer) depend on the right random initialization for training success. Hence, the circuit does not train at all for some random initializations. They seem to converge early to a local minimum.




(a)

| Quantum circuit | Number of parameters $n_{parameters}$ | Entanglement (the higher the better) | Expressibility (the lower the better) | Average training time per batch (= 100 samples) |
|---|---|---|---|---|
| **QFM ($n_{iteration}$ = 5)** | 74 | 0.79 | 0.003 | 5 min 38 s ± 8 s |
| **14 ($n_{iteration}$ = 1)** | 16 | 0.50 | 0.02 | 16 s ± 4 s |

(b)

*Figure 12. Comparison of the validation loss (a) and relevant parameters (b) of the quantum feature map approach (8 and 10 classical MLP layers), as well as circuit 14 with 10 layers.*

When comparing the validation loss of the 8 and 10-layer version using the quantum feature map approach to the quantum circuit 14 (10 classical layers) from before within the first 20 epochs, they perform quite similar. Even though the quantum feature map version exhibits favourable parameters regarding entanglement and expressibility, within the epochs displayed in Fig. 12a) they perform similar and circuit 14 seems to perform even a bit better. However, this behaviour could change when training for more epochs. Regarding the training time per batch, shown in Fig. 12b) it can be seen that circuit 14 trains much faster and completes 100 epochs within $\mathcal{O}(days)$. The quantum feature map version, in comparison, needs $\mathcal{O}(weeks)$ due to the high number of gates and thus parameters to optimize.

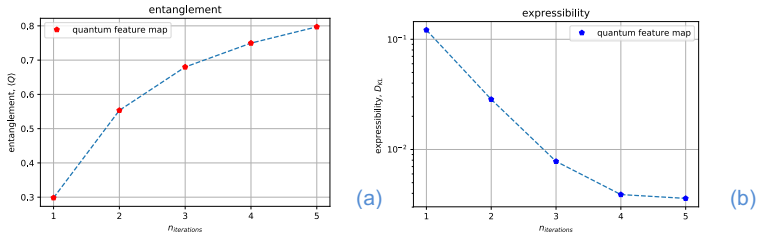



*Figure 13. Change in entanglement (a) and expressibility (b) value for quantum feature map QC for different $n_{iterations}$. Calculated as in [10].*

Also, for the quantum feature map approach, the entanglement and expressibility of the circuit increases as the number of $n_{iterations}$ increases, as shown in Fig. 13. This is accompanied by an increased training time.

## 6. DISCUSSION

Combining quantum circuits and artificial neural networks to hybrid quantum-classical neural networks is a promising and interesting field of study with possible applications using noisy intermediate-scale quantum technologies. Due to the long simulation times of quantum circuits using quantum gates, the number of qubits currently used is quite limited. This leads to a reduced expressiveness of the circuits with a low number of parameters. The entanglement and expressibility values of the different tested quantum circuits can be improved by repeating their main structure and thus increase the number of parameters within the network. To test more complex circuits efficiently, it is important to speed-up the simulation process using GPUs and parallelisation techniques and hence be able to use more qubits and more training data. This would open future possibilities to replace larger parts of classical neural networks with expressive quantum circuits for embedding and classification tasks. It would also allow to increase the size of the training data sets.

## 7. REFERENCES

[1] “Technical Proposal for the Phase-II Upgrade of the CMS Detector,” 6 2015.

[2] S. Amrouche et al., “The Tracking Machine Learning challenge: Accuracy phase,” 2019. [Online]. Available: http://arxiv.org/abs/1904.06778

[3] S. Farrell et al., “Novel deep learning methods for track reconstruction.”, 2018. arXiv:1810.06111v1 and HEP.TrkX: https://heptrkx.github.io/

[4] N. Choma et al., “Track seeding and labelling with embedded-space neural networks”, 2020. arXiv:2007.00149

[5] C. Tüysüz, F. Carminati, B. Demirköz, D. Dobos, F. Fracas, K. Novotny, K. Potamianos, S. Vallecorsa, and J.-R. Vlimant, “Particle track reconstruction with quantum algorithms,” 2020.

[6] C. Tüysüz et al., “A quantum graph neural network approach to particle track reconstruction,” 2020.

[7] X. Ju et al. “Graph neural networks for particle reconstruction in high energy physics detector”, 2019. arXiv:2003.11603

[8] Nielsen, M., & Chuang, I. (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511976667

[9] David Rousseau, Sabrina Amrouche, Paolo Calafiura, Victor Estrade, Steven Farrell, et al., “The TrackML challenge.”, NIPS 2018 - 32nd Annual Conference on Neural Information Processing Systems, Dec 2018, Montreal, Canada. pp.1-23. hal-01745714. https://www.kaggle.com/c/trackml-particle-identification/data

[10] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, “Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms,” Advanced Quantum Technologies, vol. 2, no. 12, p. 1900070, Oct 2019. [Online]. Available: http://dx.doi.org/10.1002/qute.201900070

[11] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, “Quantum embeddings for machine learning,” 2020.

[12] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, 2019.

[13] Bergholm et al., “PennyLane: Automatic differentiation of hybrid quantum-classical computations.”, 2018. arXiv:1811.04968

[14] Abraham et al., “Qiskit: An Open-source Framework for Quantum Computing”, 2019.

[15] D. Kingma et al., “Adam: A Method for Stochastic Optimization”, 2015. arXiv:1412.6980

[16] values reproduced with C. Tüysüz, A. Açar as in [10].

[17] M. Cerezo and P. J. Coles, “Impact of barren plateaus on the hessian and higher order derivatives,” 2020.