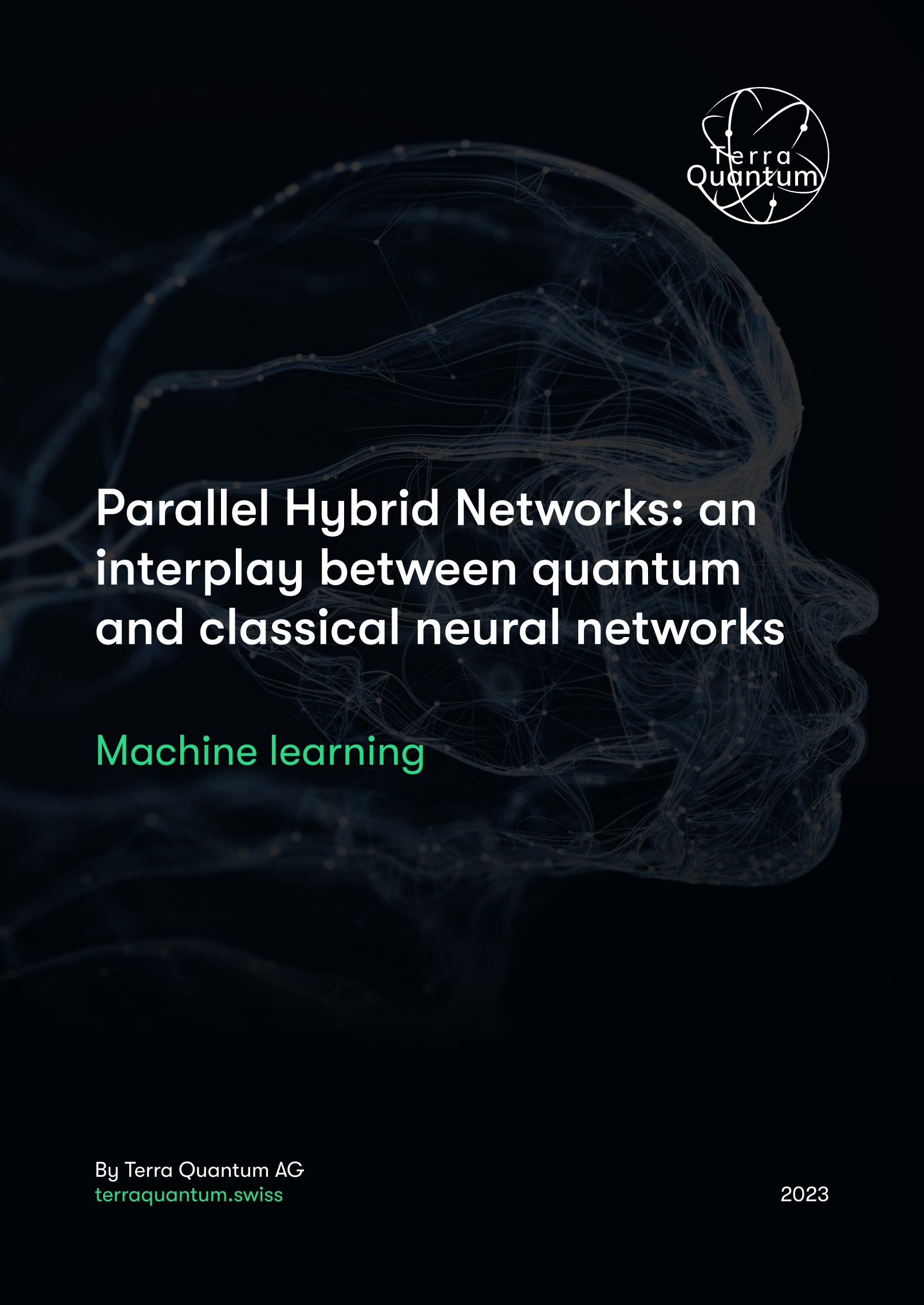

Parallel Hybrid Networks: an interplay between quantum and classical neural networks

Machine learning

By Terra Quantum AG
terraquantum.swiss

2023

Parallel Hybrid Networks: an interplay between quantum and classical neural networks

Mohammad Kordzanganeh, Daria Kosichkina, and Alexey Melnikov
Terra Quantum AG, Kornhausstrasse 25, 9000 St. Gallen, Switzerland

Quantum neural networks represent a new machine learning paradigm that has recently attracted much attention due to its potential promise. Under certain conditions, these models approximate the distribution of their dataset with a truncated Fourier series. The trigonometric nature of this fit could result in angle-embedded quantum neural networks struggling to fit the non-harmonic features in a given dataset. Moreover, the interpretability of neural networks remains a challenge. In this work, we introduce a new, interpretable class of hybrid quantum neural networks that pass the inputs of the dataset in parallel to 1) a classical multi-layered perceptron and 2) a variational quantum circuit, and then the outputs of the two are linearly combined. We observe that the quantum neural network creates a smooth sinusoidal foundation base on the training set, and then the classical perceptrons fill the non-harmonic gaps in the landscape. We demonstrate this claim on two synthetic datasets sampled from periodic distributions with added protrusions as noise. The training results indicate that the parallel hybrid network architecture could improve the solution optimality on periodic datasets with additional noise.

I. INTRODUCTION

Machine learning and quantum computing have become attractive research areas in recent years. The quest for an efficient quantum neural network (QNN) has dominated the cross-section of these two technologies. Many suggestions have been made for the potential inner workings of a classically-intractable quantum machine learning model [1, 2], but theoretical and hardware limitations could prove challenging to implement. The noise-free barren plateau problem [3] or the curse of dimensionality [4] are examples of the theoretical challenges with QNNs. At the same time, the hardware limitations point to the industry limits on the accuracy, and the number of qubits [5, 6]. Therefore, contemporary practical use of quantum technologies in machine learning should come from complementary quantum-classical architectures, called hybrid quantum neural networks (HQNN), that employ relatively small, realisable quantum circuits and classical multi-layered perceptrons (MLP) where the two work in tandem. In [7–10], we explored the applicability and performance of sequential HQNNs, where MLPs and QNNs are connected in series, passing the information from one network to another. The sequential HQNNs could introduce information bottlenecks in the representational power of the model, which could limit the expressivity of the network. This work explores the theoretical basis of parallel HQNNs, where variational quantum circuits (VQC) and MLPs process information in parallel. The approach is based on the universality theorems from two sources: 1) MLPs can produce non-harmonic functions [11] and 2) QNNs fit smooth truncated Fourier series on the training data [12]. This work was inspired by the Fourier neural operator introduced by Ref. [13].

In Sec II, we review the theoretical foundations of MLP and VQCs, and in Sec III, we introduce the design and experimental results of PHN. In Sec III B, we address the potential problem of component primacy in training PHNs, where either the VQC or the MLP could dominate the training, and propose a remedy to it. Finally, in Sec IV, we summarise our findings and discuss future directions.

II. THEORETICAL FOUNDATION

In this study, we concentrate on solving a supervised regression problem using a dataset $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathcal{X}$ is a feature vector and $y_i \in \mathcal{Y}$ is the label. Our objective is to discover a function $f(\mathbf{x})$ that can approximate the labels y of out-of-sample features. To achieve this, we create a machine learning model with parameters θ to create the functionality, $f\theta(\mathbf{x})$. We adjust these parameters according to the training sample to maximise the probability of obtaining the correct label for a given feature. The general functionality f is a machine learning architecture, while a specific realisation of its parameters, θ, is a machine learning model. In the subsequent sections, we will explore two well-known architectures and then use that theoretical foundation to justify the PHN architecture in Sec III.

A. Multi-layered perceptrons

MLPs constitute a large class of successful machine learning architectures. They are directional graphs whose nodes are ordered in one-dimensional layers which take input from the previous layer and provide the next layer with their outputs. In the case of fully connected MLPs (FCN), all neurons of each layer feed information to all the neurons in their immediate front neighbourhood. Each edge of the graph has an associated multiplicative factor (weight), and each neuron has an associated additive quantity (bias), which together form the parameters of the MLP. The first neural layer is called

the input layer, and the last is the logits. Ref. [14] provides a comprehensive overview of neural networks and their properties.

Ref. [15] showed that MLPs are asymptotically universal approximators whose fit on the training data becomes perfect as the numbers of neurons in the intermediate neural layers approach infinity. Moreover, [11, 16] proved this using a novel graphical method that showed a fully-connected network with a single intermediate neural layer can approximate any function by fitting a superposition of rectangular waves. To utilise this graph as a machine learning architecture, we could encode the features of a data point taken from the sample, $\mathbf{x}_i$, onto the input layer and then propagate their values through the graph by multiplying their values by the weights of the architecture $\mathbf{w}$ and adding the biases $\mathbf{b}$

$$h_i = \sigma(w_{i,j}^{(0)} x_j + b_i), \tag{1}$$

where σ is known as the activation function[1], h_i indicates the values of the consecutive neural layer, and Einstein's summation notation is implied. The propagation process can be passed along to the entire graph until we arrive at the terminal nodes (in this case, for a single label, we require a single terminal node). The terminal node is the output of the function and is, therefore, the approximation of the model to the label associated with the input features. We can then employ an optimisation strategy such as the stochastic gradient descent [17] to improve this approximation by adjusting the weights and biases.

B. Variational quantum circuits

Variational quantum circuits (VQC) employ variational group rotations to create a machine learning architecture on a quantum computer [18–21]. To construct a VQC, one could start by creating a quantum node of several qubits in the ground state. Then, a series of variational and fixed quantum gates can be applied to the circuit. The variational gates could include the Pauli rotation gates, which require single-qubit time evolution Hamiltonians of the respective Pauli gate. The fixed quantum gates might consist of the controlled-NOT (CNOT) and the Hadamard (H) gates. We split the variational gates into embedding, and trainable gates, which encode the features, $\mathbf{x}$, and act as model parameters, θ. At the end of the circuit, we measure the qubits in a specified basis, such as the Pauli bases, and obtain either a 0 or a 1. After many iterations of the circuit, we can find the likelihood of getting a 0 over 1, and by taking the average, we can obtain the expectation value of the circuit. By the Born rule [22], we can find this probability by taking the expectation of the measurement matrix, M, and then using it as the output of our model:

$$f(\mathbf{x}, \theta) = \langle \psi(\mathbf{x}, \theta) | \, M \, | \psi(\mathbf{x}, \theta) \rangle \,, \tag{2}$$

where $|\psi(\mathbf{x}, \theta)\rangle$ denotes the state of the quantum circuit before the measurement. We can improve this approximation to the labels by optimising the parameters of the VQC, θ. [12] proved that VQCs are also universal approximators, and the way they work is by fitting a truncated Fourier series over the samples:

$$f(x) = \sum_{k=-L}^{L} c_k e^{ikx}, \tag{3}$$

where L is the highest degree Fourier term expressible by the VQC.

III. RESULTS – PARALLEL HYBRID NETWORKS

We split the HQNN hybrid interfaces into two categories: 1) sequential: where the classical and quantum parts feed directly into each other, and 2) parallel: where a classical multi-layered perceptron and a variational quantum circuit in parallel process the same information. In this section, we take an in-depth look into HQNNs of the latter type and the functions they represent. We shall refer to these networks as parallel hybrid networks (PHN). Fig 1 shows the general architecture of PHNs. The combination is a weighted linear addition with trainable weights. These weights determine the contribution of each network to the final output. The specific VQC used here is a generalised data re-uploading VQC, where K qubits are initialised in the state $|0\rangle^{\otimes K}$. Then in alternation, a series of variational and encoding layers are applied. The encoding layers S take the input features, $\{x_1, \cdots, x_N\}$, and encode them in a unitary transformation which is then applied to the state of the qubit. The variational layers, U, are unitaries that encapsulate the VQC model parameters as an operator that can be used for the quantum state of the network. Finally, the measurements are where the quantum information collapses into M classical outputs, which can be obtained by taking the expectation value of the circuit with respect to the measurement observable. Note the difference between M, the number of classical outputs out of the VQC, and K, the number of qubits, and that they are not necessarily the same, as often we are only required to measure some of the qubits.

In parallel, the fully connected MLP also takes in the N features and passes them to a single layer of hidden neurons of size F by multiplying the feature vector by a

[1] In this work, the particular functionality of the activation function is not material, and for simplicity, we take it to be the sigmoid function everywhere, $\sigma(t) = \frac{1}{1+e^{-t}}$, and otherwise only where clearly stated.

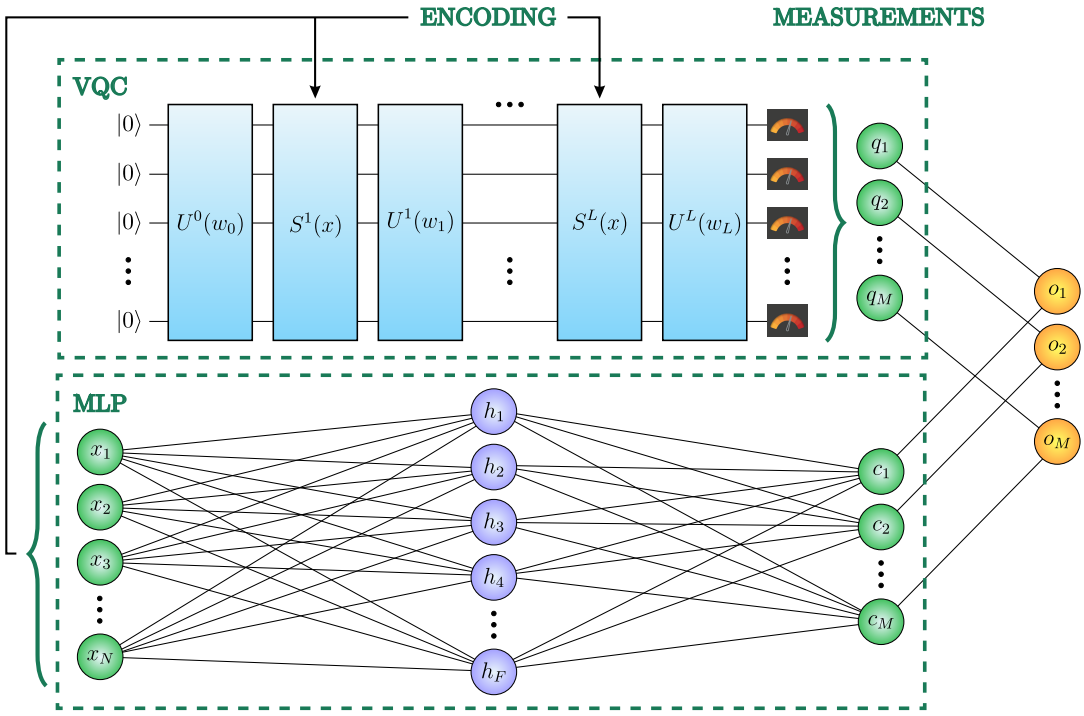

FIG. 1: The general architecture of the PHN. The PHN takes an input vector of features and passes them to an angle-embedded VQC (a VQC that uses single-qubit, Pauli gate embedding of features without applying any non-linear kernel on the features) of the appropriate architecture in parallel to a multi-layered perceptron with a single hidden layer of the appropriate size. The outputs of the VQC are then combined linearly with the outputs of the MLP to produce a final output vector.

weight matrix of size $N \times F$. Then, biases are applied to these values and scaled using an activation function. The use of an activation function is necessary as it provides a measure of non-linearity to an otherwise linear system. Then, the neurons are propagated to M MLP output neurons with their own biases and activation functions, denoted as $\{c_1, \cdots, c_M\}$. The MLP and VQC outputs are then combined, using a two-to-one linear weight layer, to form the PHN outputs, $\{o_1, \cdots, o_M\}$. This final layer combines the first output of the VQC with the first output of the MLP: $o_1 = s_1^c c_1 + s_1^q q_1$, and similarly for all the M outputs, where $(\{s^q\}, \{s^c\})$ are trainable parameters.

In Sec II, we saw that an MLP with a single hidden layer created a non-harmonic functional fit for the dataset and that the VQC created a truncated Fourier series, a harmonic function. Thus, a network combining these two results could map the smooth, sinusoidal parts through the VQC and fill the protruding sections via the MLP. This complementary setting has the potential to approximate a function that fits the dataset both in the position space (MLP) and in the conjugate momentum space (VQC). We could compare this duality to the Fourier neural operator in Ref. [13] or the models with benign overfitting in Ref. [23]. The scope of this work includes architectures that use multiple VQCs (MLPs) in parallel, as they can always be combined to form a single VQC (MLP).

A. Performance

We start with a ground truth consisting of an overall single-frequency sinusoidal function and then introduce high-frequency perturbations to this system. Specifically, the functional form was

$$f = \sin(x) + 0.05\sin(8x) + 0.03\sin(16x) + 0.01\sin(32x),$$

which was scaled to -1 and 1. 100 equally-spaced data samples were taken from this distribution for training. We train a simple PHN, described in detail in Appendix A, to recreate the ground truth as accurately as possible. We then train the individual constituents of the same PHN architecture to see their performances. Fig 2 shows the training loss curves, and Fig 3 shows the best fits that each architecture created for the ground truth. The PHN trains to a lower MSE training loss than its elements, which suggests that adding the VQC improves the overall expressivity of the MLP. Furthermore, by examining the loss curves, we see that the PHN inherits the same speedy descent as the VQC but also shares many of the features present in the MLP loss curve, such as the spikes or the gradual flattening near the end of training.

We see that the PHN outperforms both individual components, which means that both the VQC and MLP contribute to the training, and neither becomes redundant. In Sec III B, we explore how to measure the contri-

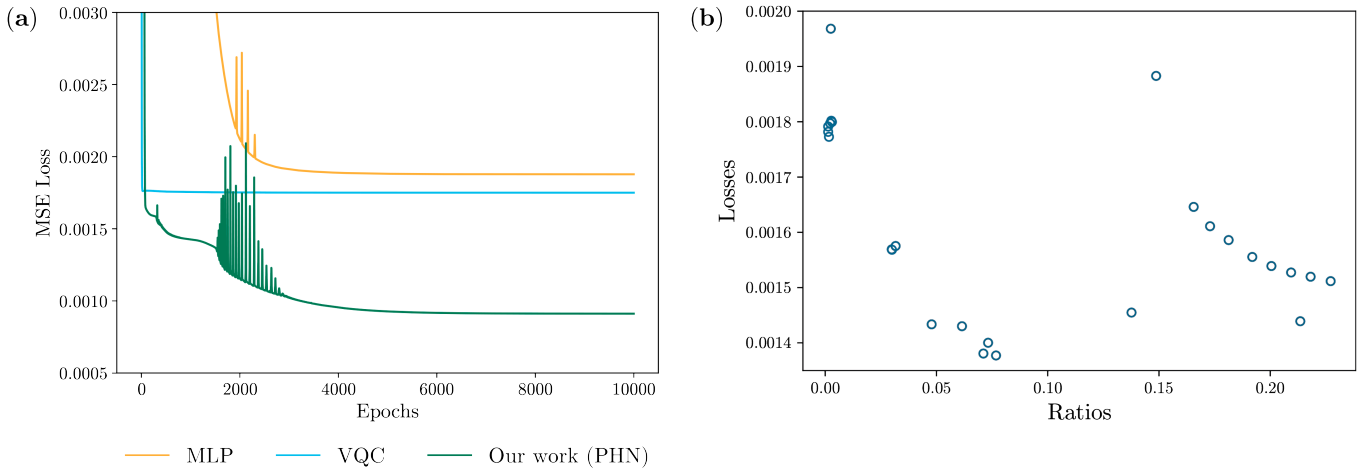

FIG. 2: (a) The training losses of the individual elements of the PHN when trained separately, as well as the full PHN. (b) The scatter plot of final losses and their respective final ratios after 1000 epochs of training. The optimal loss value is achieved at non-zero ratios, where ratios to the side of this value provide sub-optimal losses. Note that this figure only includes the runs with learning rates whose final loss is low enough for comparison.

bution of each and how the tuning of hyper-parameters could change this contribution.

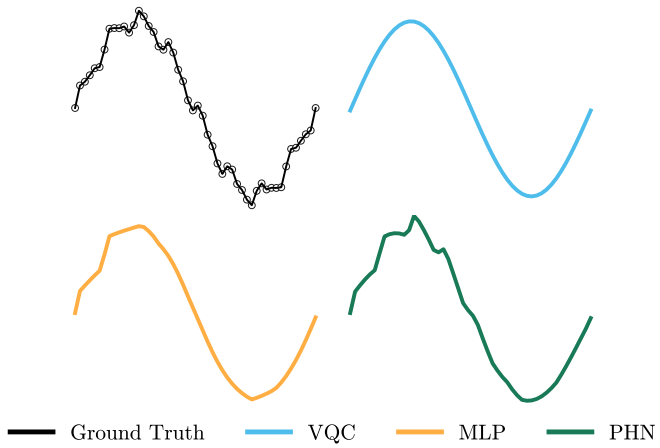

FIG. 3: The functional fits of each architecture to the ground truth. The VQC, expectedly, produced a sinusoidal curve. The MLP created an overall curve close to a sinusoidal curve but with jagged edges. The PHN, however, produced the best result, predicting the protrusion at the peak.

B. PHN primacy

A way to understand the relative contributions of each network is by inspecting the weights of the combination phase, s_q and s_c, for VQC and MLP, respectively. In this section, we look at how this contribution can unfold when training the PHN.

When training a PHN, we must be wary that the VQC and MLP train at different rates. We define the primacy of one of the constituent architectures (VQC or MLP) over another as when the last weights preceding the PHN output layer vanish for one of the components. Equivalently, this makes the output of the latter network independent from the input features, which would mean that the prediction curve is solely constructed by either the MLP or the VQC. A primacy of this type could prevent the PHN from reaching the global minimum, as it is limited to what only one of the components could offer.

The ratio of the final weights, $r = \frac{|s_c|}{|s_q|}$, was used to track intervals of different primacy regimes recorded for different hyper-parameterisations of the PHN. Specifically, we fixed the learning rate of the VQC at 0.01 and then selected the learning rate of the MLP from 54 values of $\alpha_c \in \{1.0\text{e}{-7}, 2.0\text{e}{-7}, \cdots, 9.0\text{e}{-7}, 1.0\text{e}{-6}, \cdots, 9.0\text{e}{-2}\}$. We, then, trained the PHN for 1000 epochs at a fixed initialisation point. Lastly, we recorded the ratios r throughout each training. The bigger the ratio, the more the MLP would contribute compared with the VQC. Notably, even a small contribution could make a critical difference, and primacy occurs only when one of the contributions completely vanishes.

The dependence of the final loss on the ratio of the final weights for the VQC and MLP, shown in Fig 2(b), highlights the potential for either component to dominate training in the PHN architecture. The results exhibit an optimal range for the ratio between 0.1 and 1, indicating that a balanced contribution from the VQC and MLP is desirable for achieving the best results. It is also evident that complete MLP primacy, where the ratio approaches 0, leads to worse final losses. However, we also observe that adjusting the learning rates of the two components can sometimes improve the loss. Therefore, tuning the learning rates of the VQC and MLP is crucial to achieve a balanced contribution from both parts and to prevent either component from dominating the training.

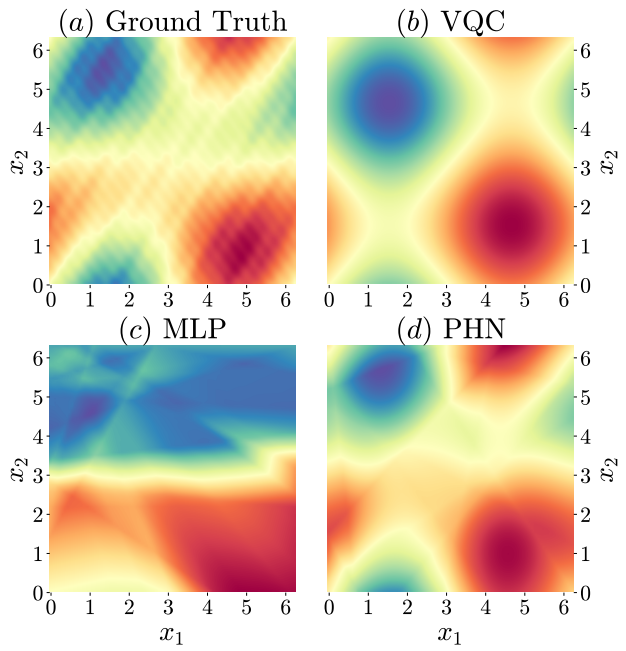

FIG. 4: Figure (**a**) shows the ground truth for our 2D problem in the form of a contour map. The predictions shown are for the VQC (**b**), MLP (**c**), and PHN (**d**). We see that the prediction of the VQC is smooth and convex, whereas the MLP creates jagged shapes. Taking advantage of both of these properties, the PHN represents the harmonic functions of the VQC with the added, necessary protrusions.

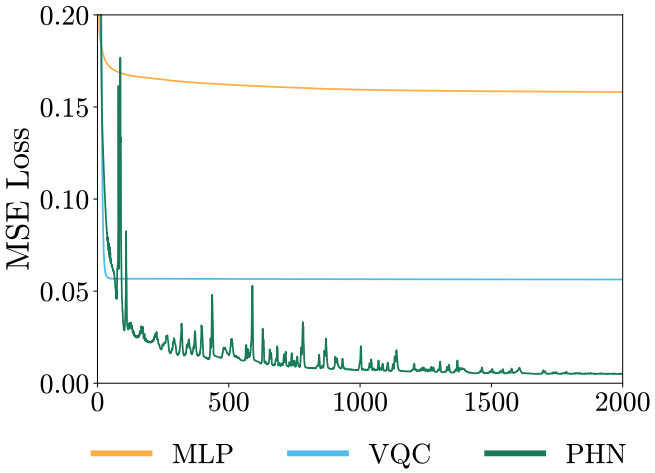

FIG. 5: The evolution of the loss function of the PHN model in Fig 6(b) and its constituents (VQC and MLP) on the 2D dataset. The PHN achieves an impressively low training MSE loss.

C. Scalability and generalisation

To show the scalability of the PHN, in this section, we try a 2-dimensional problem with the view that this can be scaled to an arbitrarily complex problem with many qubits. To solve the problem in Fig. 4(a), a simple PHN, described in Appendix. A, was employed. The distribution used to create this ground truth was

$$\begin{aligned} f(x_1, x_2) = \sin(x_1) + \sin(x_2) + 0.8\sin(x_1 + x_2) \\ +0.3\sin(x_1 - x_2) + 0.09\sin(8x_1 + 4x_2) \\ +0.05\sin(16x_1 - 12x_2) + 0.04\sin(12x_1 + 8x_2). \end{aligned}$$

Note that this function (similarly to the 1D case) was chosen entirely at random to have a coarse harmonic structure (first four terms) as well as high-frequency noise (the last three terms) and not engineered to showcase the PHN in a favourable light. 100 equidistant points were sampled from this ground truth to create a training set. This set was then trained on only the VQC, MLP, and the complete PHN for 10,000 epochs. The trained models were then tested on 10,000 equidistant points data points to see the generalisation ability of each architecture. Figs. 4(b), (c), and (d) respectively showcase the fit of the VQC, MLP, and PHN, and Fig 5 shows the evolution of their training loss. We see that the VQC creates a symmetric, sinusoidal pattern, whereas the MLP creates jagged regions to fit the ground truth. However, the PHN can generalise the ground truth by employing both elements and thus creates a closer fit, which could mean that for such datasets, the PHN could provide a high generalisation power over the MLP or the VQC.

IV. CONCLUSION

Overall, our findings demonstrate the potential of PHN as a powerful tool for quantum machine learning. It is a hybrid architecture that can extract harmonic and non-harmonic features from a dataset. By leveraging its unique architecture, the PHN can learn complex patterns and relationships within the data that might be difficult to capture using traditional machine learning algorithms.

However, it is essential to note that the performance of the PHN is highly dependent on the choice of hyperparameters. The number of layers, number of neurons in each layer, activation functions, and learning rate are crucial in determining how well the network performs on a given task. Therefore, hyperparameter tuning is a critical step in training a successful PHN. One potential direction for future research is to explore using a custom learning rate scheduler to modify the learning rate during training. A learning rate scheduler can dynamically adjust the learning rate based on the network's performance on the training set, allowing the model to learn more efficiently and converge faster. Implementing a learning rate scheduler may further improve the performance of the PHN on a wide range of tasks.

[1] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

[2] Matthias C. Caro, Hsin-Yuan Huang, M. Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J. Coles. Generalization in quantum machine learning from few training data. *arXiv:2111.05292*, 2021.

[3] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[4] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):2631, 2021.

[5] Mohammad Kordzanganeh, Markus Buchberger, Maxim Povolotskii, Wilhelm Fischer, Andrii Kurkin, Wilfrid Somogyi, Asel Sagingalieva, Markus Pflitsch, and Alexey Melnikov. Benchmarking simulated and physical quantum processing units using quantum and hybrid algorithms. *arXiv:2211.15631*, 2022.

[6] Samson Wang, Enrico Fontana, Marco Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, 12(1):6961, 2021.

[7] Michael Perelshtein, Asel Sagingalieva, Karan Pinto, Vishal Shete, Alexey Pakhomchik, Artem Melnikov, Florian Neukart, Georg Gesek, Alexey Melnikov, and Valerii Vinokur. Practical application-specific advantage through hybrid quantum computing. *arXiv:2205.04858*, 2022.

[8] Asel Sagingalieva, Andrii Kurkin, Artem Melnikov, Daniil Kuhmistrov, Michael Perelshtein, Alexey Melnikov, Andrea Skolik, and David Von Dollen. Hyperparameter optimization of hybrid quantum neural networks for car classification. *arXiv:2205.04878*, 2022.

[9] Asel Sagingalieva, Mohammad Kordzanganeh, Nurbolat Kenbayev, Daria Kosichkina, Tatiana Tomashuk, and Alexey Melnikov. Hybrid quantum neural network for drug response prediction. *arXiv:2211.05777*, 2022.

[10] Serge Rainjonneau, Igor Tokarev, Sergei Iudin, Saaketh Rayaprolu, Karan Pinto, Daria Lemtiuzhnikova, Miras Koblan, Egor Barashov, Mohammad Kordzanganeh, Markus Pflitsch, et al. Quantum algorithms applied to satellite mission planning for Earth observation. *arXiv:2302.07181*, 2023.

[11] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[12] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A*, 103:032430, 2021.

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv:2010.08895*, 2020.

[14] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[16] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[17] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[18] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[19] Alexey Melnikov, Mohammad Kordzanganeh, Alexander Alodjants, and Ray-Kuang Lee. Quantum machine learning: from physics to software engineering. *Advances in Physics: X*, 8(1):2165452, 2023.

[20] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4:043001, 2019.

[21] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

[22] Max Born. Zur Quantenmechanik der Stoßvorgänge. *Zeitschrift fur Physik*, 37:863–867, 1926.

[23] Evan Peters and Maria Schuld. Generalization despite overfitting in quantum machine learning models. *arXiv:2209.05523*, 2022.

Appendix A: The exact experimental setup

Figs 6(a) and 6(b) illustrate the PHN example architectures used to produce the 1D and 2D results, respectively. In both cases, the quantum measurement in q_{out} was made in the Z-basis, and the MLP utilised a single hidden layer with the rectified linear unit (ReLU) and sigmoid activation functions for the hidden and output layers, c_{out}, respectively. The MLP was fully connected and included weights and biases. The outputs of the MLP and VQC were linearly combined after being weighted by s_c and s_q, respectively. The MLP had 769, the VQC had 3, and the final weighing layer had two parameters.

Fig 6(b) depicts a simple 2-dimensional PHN used to demonstrate the scalability of the PHN. The activation layers employed in the MLP were ReLU and sigmoid for the first and second layers. The VQC produced a single output, q_1, which resulted from measuring the state of the VQC in the $Z \bigotimes I$ basis, where the identity measurement I is excluded from the diagram. A learning rate of 0.01 was used for the VQC parameters and 0.001 for all others. We also utilised the Adam optimiser and a learning rate scheduler that multiplied all learning rates by $\gamma = 0.99$ every ten epochs.

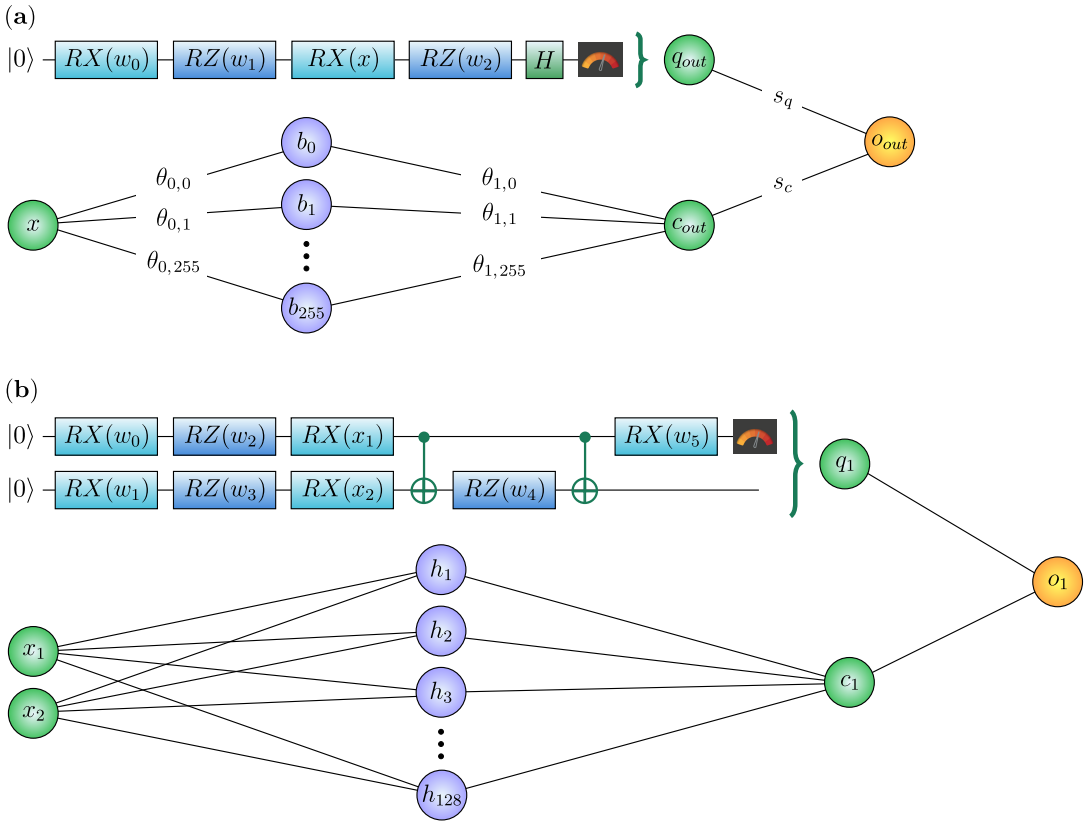

FIG. 6: (a) The schematic diagram describes the joint work of the VQC with one qubit and a basic MLP architecture. (b) The architecture includes a two-qubit VQC with only a measurement applied to the first qubit and an MLP with 128 neurons in its singular hidden layer.