RESEARCH ARTICLE

Hybrid quantum ResNet for car classification and its hyperparameter optimization

Asel Sagingalieva[1] · Mo Kordzanganeh[1] · Andrii Kurkin[1] · Artem Melnikov[1] · Daniil Kuhmistrov[1] ·
Michael Perelshtein[1] · Alexey Melnikov[1] · Andrea Skolik[2,4] · David Von Dollen[3,4]

Abstract
Image recognition is one of the primary applications of machine learning algorithms. Nevertheless, machine learning models used in modern image recognition systems consist of millions of parameters that usually require significant computational time to be adjusted. Moreover, adjustment of model hyperparameters leads to additional overhead. Because of this, new developments in machine learning models and hyperparameter optimization techniques are required. This paper presents a quantum-inspired hyperparameter optimization technique and a hybrid quantum-classical machine learning model for supervised learning. We benchmark our hyperparameter optimization method over standard black-box objective functions and observe performance improvements in the form of reduced expected run times and fitness in response to the growth in the size of the search space. We test our approaches in a car image classification task and demonstrate a full-scale implementation of the hybrid quantum ResNet model with the tensor train hyperparameter optimization. Our tests show a qualitative and quantitative advantage over the corresponding standard classical tabular grid search approach used with a deep neural network ResNet34. A classification accuracy of 0.97 was obtained by the hybrid model after 18 iterations, whereas the classical model achieved an accuracy of 0.92 after 75 iterations.

1 Introduction

The field of quantum computing has seen large leaps in building usable quantum hardware during the past decade. As one of the first vendors, D-Wave provided access to a quantum device that can solve specific types of optimization problems Johnson et al. (2011). Motivated by this, quantum computing has not only received much attention in the research community, but was also started to be perceived as a valuable technology in industry. Volkswagen published a pioneering result on using the D-Wave quantum annealer to optimize traffic flow in 2017 Neukart et al.

(2017), which prompted a number of works by other automotive companies Mehta et al. (2019); Ohzeki et al. (2019); Yarkoni et al. (2021). Since then, quantum annealing has been applied in a number of industry-related problems like chemistry Streif et al. (2019); Xia et al. (2017), aviation Stollenwerk et al. (2019), logistics Feld et al. (2019), and finance Grant et al. (2021). Aside from quantum annealing, gate-based quantum devices have gained increased popularity, not least after the first demonstration of a quantum device outperforming its classical counterparts Arute et al. (2019). A number of industry-motivated works have since been published in the three main application areas that are currently of interest for gate-based quantum computing: optimization Streif et al. (2021); Streif and Leib (2020); Amaro et al. (2022); Dalyac et al. (2021); Luckow et al. (2021), quantum chemistry and simulation Arute et al. (2020); Malone et al. (2022), and machine learning Melnikov et al. (2023); Rudolph et al. (2020); Skolik et al. (2021, 2022); Peters et al. (2021); Alcazar et al. (2020); Perelshtein et al. (2022); Sagingalieva et al. (2022); Kordzanganeh et al. (2022). Research in the industrial context has been largely motivated by noisy

✉ Alexey Melnikov
ame@terraquantum.swiss

[1] Terra Quantum AG, St. Gallen 9000, Switzerland

[2] Volkswagen Data:Lab, Munich 80805, Germany

[3] Volkswagen Group of America, Auburn Hills, MI 48326, USA

[4] Leiden University, Leiden, CA 2333, The Netherlands

intermediate-scale quantum (NISQ) devices Kordzanganeh et al. (2022)—early quantum devices with a small number of qubits and no error correction. In this regime, variational quantum algorithms (VQAs) have been identified as the most promising candidate for near-term advantage due to their robustness to noise Cerezo et al. (2021). In a VQA, a parametrized quantum circuit (PQC) is optimized by a classical outer loop to solve a specific task like finding the ground state of a given Hamiltonian or classifying data based on given input features. As qubit numbers are expected to stay relatively low within the next years, hybrid alternatives to models realized purely by PQCs have been explored Zhang et al. (2021); Mari et al. (2020); Zhao and Gao (2019); Dou et al. (2021); Sebastianelli et al. (2021); Pramanik et al. (2021); Perelshtein et al. (2022); Rainjonneau et al. (2023); Sagingalieva et al. (2022). In these works, a quantum model is combined with a classical model and optimized end-to-end to solve a specific task. In the context of machine learning, this means that a PQC and neural network (NN) are trained together as one model, where the NN can be placed either before or after the PQC in the chain of execution. When the NN comes first, it can act as a dimensionality reduction technique for the quantum model, which can then be implemented with relatively few qubits.

In this work, we use a hybrid quantum ResNet model to perform image classification on a subset of the Stanford Cars dataset Krause et al. (2013). Image classification is an ubiquitous problem in the automotive industry and can be used for tasks like sorting out parts with defects. Supervised learning algorithms for classification have also been extensively studied in quantum literature Havlíček et al. (2019); Schuld and Killoran (2019); Schuld et al. (2020); Rebentrost et al. (2014), and it has been proven that there exist specific learning tasks based on the discrete logarithm problem where a separation between quantum and classical learners exists for classification Liu et al. (2021). While the separation in Liu et al. (2021) is based on Shor's algorithm and therefore not expected to transfer to realistic learning tasks as the car classification mentioned above, it motivates further experimental study of quantum-enhanced models for classification on real-world datasets.

In combining PQCs and classical NNs into hybrid quantum-classical models, we encounter a challenge in searching hyperparameter configurations that produce performance gains in terms of model accuracy and training. Hyperparameters can be considered values that are set for the model and do not change during the training regime and may include variables such as learning rate, decay rates, choice of optimizer for the model, number of qubits, or layer sizes. Often in practice, these parameters are selected by experts based upon some a priori knowledge and trial-and-error. This limits the search space, but in turn can lead to producing a suboptimal model configuration.
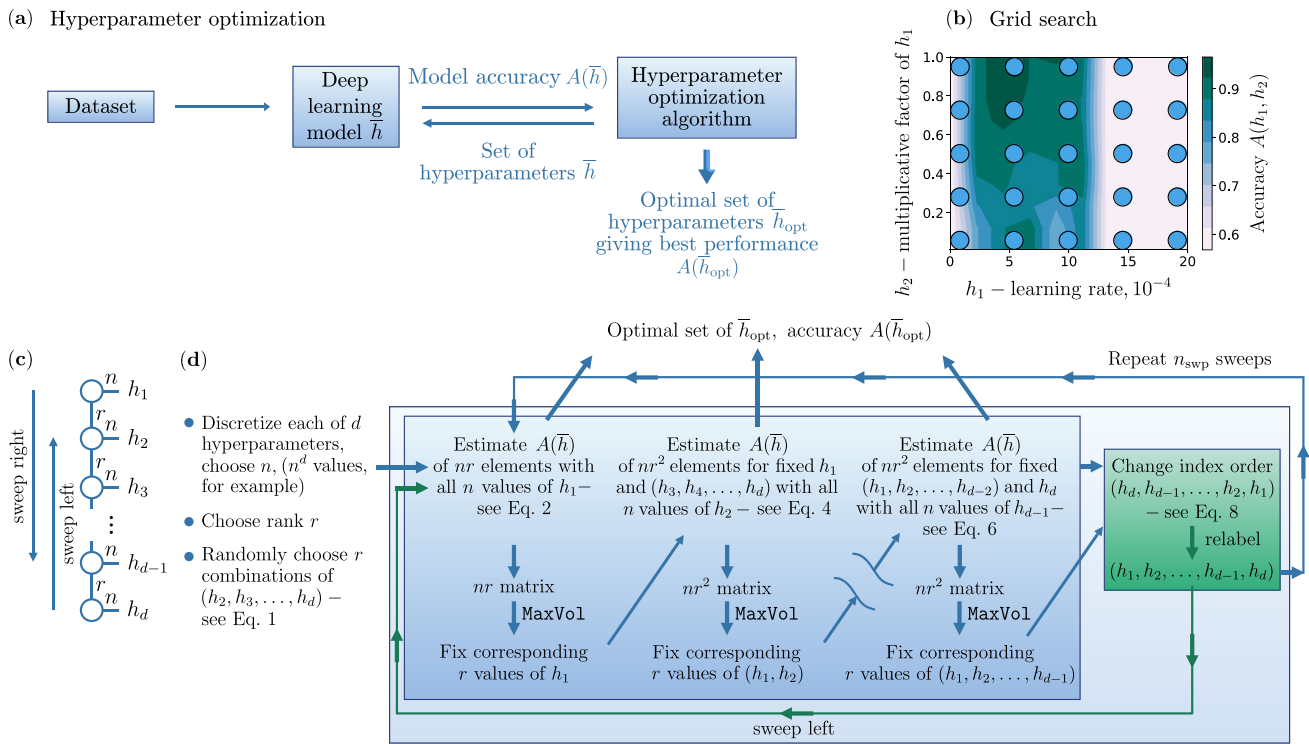
Hyperparameter optimization is the process of automating the search for the best set of hyperparameters, reducing the need for expert knowledge in hyperparameter configurations for models, with an increase in computation required to evaluate configurations of models in search of an optimum. In the 1990s, researchers reported performance gains leveraging a wrapper method, which tuned parameters for specific models and datasets using best-first search and cross validation Kohavi et al. (1995). In more recent years, researchers have proposed search algorithms using bandits Li et al. (2017), which leverage early stopping methods. Successive halving algorithms such as the one introduced in Karnin et al. (2013) and the parallelized version introduced in Li et al. (2018) allocate more resources to more promising configurations. Sequential model-based optimization leverages Bayesian optimization with an aggressive dual racing mechanism and also has shown performance improvements for hyperparameter optimization Hutter et al. (2011); Lindauer and Hutter (2018). Evolutionary and population-based heuristics for black-box optimization have also achieved state-of-the-art results when applied to hyperparameter optimization in numerous competitions for black-box optimization Vermetten et al. (2020); Bäck (1996); Awad et al. (2020). In recent years, a whole field has formed around automating the process of finding optimal hyperparameters for machine learning models, with some prime examples being neural architecture search Elsken et al. (2019) and automated machine learning (AutoML) Hutter et al. (2019). Automating the search of hyperparameters in a quantum machine learning (QML) context has also started to attract attention, and the authors of Gómez et al. (2022) have explored the first version of AutoQML.

Our contribution in this paper is not only to examine the performance gains of hybrid quantum ResNet models vs. purely classical, but also to investigate whether quantum-enhanced or quantum-inspired methods may offer an advantage in automating the search over the configuration space of the models. We show a reduction in computational complexity in regard to expected run times and evaluations for various configurations of models, the high cost of which motivate this investigation. We investigate using the tensor train decomposition for searching the hyperparameter space of the hybrid quantum neural network (HQNN) framed as a global optimization problem as in Zheltkov and Osinsky (2020). This method has been successful in optimizing models of social networks in Kabanikhin et al. (2019) and as a method of compression for deep neural networks Wang et al. (2021).

2 Results

2.1 Hyperparameter optimization

The problem of hyperparameter optimization (HPO) is described schematically in Fig. 1(a). Given a certain dataset

Fig. 1 The hyperparameter optimization problem description (a). The tabular methods for hyperparameter optimization: the grid search algorithm (b) and the tensor train algorithm (c–d)

and a machine learning (ML) model, the learning model demonstrates an accuracy $A(\bar{h})$ which depends on the hyperparameters $\bar{h}$. To achieve the best possible model accuracy, one has to optimize the hyperparameters. To perform the HPO, an unknown black-box function $A(\bar{h})$ has to be explored. The exploration is an iterative process, where at each iteration the HPO algorithm provides a set of hyperparameters $\bar{h}$ and receives the corresponding model accuracy $A(\bar{h})$. As a result of this iterative process, the HPO algorithm outputs the best achieved performance $A(\bar{h}_{\mathrm{opt}})$ with the corresponding hyperparameters $\bar{h}_{\mathrm{opt}}$.

The HPO could be organized in different ways. One of the standard methods for HPO is a tabular method of grid search (GS), also known as a parameter sweep (Fig. 1(b)). To illustrate how a grid search works, we have chosen two hyperparameters: the learning rate (h_1) and the multiplicative factor of learning rate (h_2). They are plotted along the x-axis and the y-axis, respectively. The color on the contour shows the accuracy of the model $A(h_1, h_2)$ with two given hyperparameters changing from light pink (the lowest accuracy) to dark green (the highest accuracy). In the GS method, the hyperparameter values are discretized, which results in a grid of values shown as big dots. The GS algorithm goes through all the values from this grid with the goal of finding the maximum accuracy. As one can see in this figure, there are only three points at which this method can find a high accuracy with 25 iterations (shown as 25 points in Fig. 1(b)).

This example shows that there could be a better tabular HPO in terms of the best achievable accuracy and the number of iterations used.

2.2 Tensor train approach to hyperparameter optimization

Here, we propose a quantum-inspired approach to hyperparameter optimization based on the tensor train (TT) programming. The TT approach was initially introduced in the context of quantum many-body system analysis, e.g., for finding a ground state with minimal energy of multi-particle Hamiltonians via density matrix renormalization groups White (1992). In this approach, the ground state is represented in the TT format, often referred to as the matrix product state in physics Cirac et al. (2021). We employ the TT representation (shown in Fig. 1(c)) in another way here and use it for the hyperparameter optimization. As one can see in Fig. 1(c), the TT is represented as a multiplication of tensors, where an individual tensor is shown as a circle with the number of "legs" that corresponds to the rank of the tensor. h_1 and h_d circles are the matrices of $n \times r$ dimension, and $\{h_i\}_{i=2}^{i=d-1}$ is a rank 3 tensor of dimensions $n \times r^2$. The two arrows in the Fig. 1(c) illustrate sweeps right and left along with the TT. This refers to the algorithm described below. Leveraging the locality of the problem, i.e., a small correlation between hyperparameters, we perform the black-box

optimization based on the cross-approximation technique applied for tensors Oseledets and Tyrtyshnikov (2010); Zheltkov and Tyrtyshnikov (2020).

Similar to the previously discussed GS method, we discretize the hyperparameter space with TT optimization (TetraOpt) and then consider a tensor composed of scores that can be estimated by running an ML model with a corresponding set of hyperparameters. However, compared to GS, the TT method is dynamic, which means that the next set of evaluating points in the hyperparameter space is chosen based on the knowledge accumulated during all previous evaluations. With TetraOpt, we will not estimate all the scores $A(\bar{h})$ available to the model. Instead of this, we will approximate $A(\bar{h})$ via TT, referring to a limited number of tensor elements using the cross-approximation method Oseledets and Tyrtyshnikov (2010). During the process, new sets of hyperparameters for which the model needs to be evaluated are determined using the MaxVol routine Goreinov and Oseledets (2010). The MaxVol routine is an algorithm that finds an $r \times r$ submatrix of maximum volume, i.e., a square matrix with a maximum determinant module in an $n \times r$ matrix.

Hyperparameters are changed in an iterative process, in which one is likely to find a better accuracy $A(\bar{h})$ after each iteration and thus find a good set of hyperparameters. Notably, the TetraOpt algorithm requires an estimate of $\mathcal{O}(dnr^2)$ elements and $\mathcal{O}(dnr^3)$ of calculations, where d is the number of hyperparameters, n is a number of discretization points, and r is a fixed rank. If one compares it with the GS algorithm, which requires estimation of $\mathcal{O}(n^d)$ elements, one is expected to observe practical advantages, especially with a large number of hyperparameters.

The TetraOpt algorithm for the HPO is presented as the Algorithm 1 pseudocode that also corresponds to Fig. 1(d). The TetraOpt algorithm can be described with 9 steps:

1. Suppose each of d hyperparameters is defined on some interval $h_i \in [h_i^{\min}, h_i^{\max}]$, where $i \in [1, d]$. One first discretizes each of d hyperparameters by defining n points

$$\{h_i(1), h_i(2), \ldots, h_i(n)\}_{i=1}^{i=d}.$$

2. Then, we need to choose the rank r. This choice is a trade-off between computational time and accuracy, which respectively require a small and a large rank.

3. r combinations of

$$\{h_2^1(j), h_3^1(j), \ldots, h_d^1(j)\}_{j=1}^{j=r} \tag{1}$$

are chosen.

4. In the next three steps, we implement an iterative process called the "sweep right." The first step of this iterative process is related to the first TT core evaluation:

Algorithm 1 Tensor train optimization.

```
1:  Accuracy A(h̄_opt) = 0
2:  i_swp = 1
3:  Core = 1
4:  j_swp = 1
5:  Discretize each of d hyperparameters with n points
6:  Randomly choose r combinations of (h_2, h_3, ..., h_d)
7:  while i_swp ≤ n_swp do
8:      while j_swp ≤ 2 do
9:          while Core < d do
10:             if Core == 1 then
11:                 Estimate A(h̄) of nr elements with all n values of h_1
12:                 if A(h̄_opt) < A(h̄) then
13:                     A(h̄_opt) = A(h̄)
14:                 end if
15:                 MaxVol
16:                 Fix corresponding r values of h_1
17:             else
18:                 Estimate  A(h̄)  of  nr^2  elements for fixed
                    (h_1, ..., h_Core−1) with all n values of h_Core
19:                 if A(h̄_opt) < A(h̄) then
20:                     A(h̄_opt) = A(h̄)
21:                 end if
22:                 MaxVol
23:                 Fix corresponding r values of (h_1, ..., h_Core)
24:             end if
25:             Core = Core + 1
26:         end while
27:         Change index order (h_d, ..., h_1)
28:         Relabel (h_1, ..., h_d)
29:         Core = 1
30:         j_swp = j_swp + 1
31:     end while
32:     j_swp = 1
33:     i_swp = i_swp + 1
34: end while
```

- The accuracy of nr elements is estimated with all n values of the first hyperparameter $\{h_1(i_1)\}_{i_1=1}^{i_1=n}$ and for the r combinations of $\{h_2^1(j), h_3^1(j), \ldots, h_d^1(j)\}_{j=1}^{j=r}$:

$$\{A(h_1(i_1), h_2^1(j), h_3^1(j), \ldots, \\ h_d^1(j))\}_{j=1, i_1=1}^{j=r, i_1=n}. \tag{2}$$

- In this matrix of size $n \times r$, we search for a submatrix with maximum determinant module:

$$\{A(h_1^1(i_1), h_2^1(j), h_3^1(j), h_d^1(j))\}_{j=1, i_1=1}^{j=r, i_1=r}. \tag{3}$$

The corresponding r values of the first hyperparameter are fixed $\{h_1^1(i_1)\}_{i_1=1}^{i_1=r}$.

5. The next step of this iterative process is related to the second TT core evaluation:

- We fix r values $\{h_1^1(i_1)\}_{i_1=1}^{i_1=r}$ of the previous step as well as r combinations $\{h_3^1(j), h_4^1(j), \ldots, h_d^1(j)\}_{j=1}^{j=r}$ of the third step. We, then, estimate the accuracy of the

nr^2 elements with all n values of the second hyper-parameter $\{h_2(i_2)\}_{i_2=1}^{i_2=n}$:

$$\{A(h_1^1(i_1), h_2^1(i_2), h_3^1(j), \dots, \\ h_d^1(j))\}_{j=1, i_1=1, i_2=1}^{j=r, i_1=r, i_2=n} \text{.} \tag{4}$$

- Again, in this matrix of size $nr \times r$, we search for a submatrix with the maximum determinant module:

$$\{A((h_1^2(k), h_2^2(k)), h_3^1(j), \dots, \\ h_d^1(j))\}_{j=1, k=1}^{j=r, k=r} \tag{5}$$

r combinations $\{(h_1^2(k), h_2^2(k))\}_{k=1}^{k=r}$ of the first and the second hyperparameters are fixed.

6. The $d-1$ TT core evaluation:

- We fix r combinations $\{(h_1^{d-2}(k), h_2^{d-2}(k), \dots, h_{d-2}^{d-2}(k))\}_{k=1}^{k=r}$ of the $d-2$ TT core as well as r combinations $\{h_d^1(j)\}_{j=1}^{j=r}$ of the third step. We, then, estimate the accuracy of the nr^2 elements with all n values of the $\{h_{d-1}(i_d)\}_{i_d=1}^{i_d=n}$:

$$\{A((h_1^{d-2}(k), \dots, h_{d-2}^{d-2}(k)), \\ h_{d-1}(i_{d-1}), h_d^1(j))\}_{k=1, i_{d-1}=1, j=1}^{k=r, i_{d-1}=n, j=r} \tag{6}$$

- Again, in this matrix of size $nr \times r$, we search for a submatrix with the maximum determinant module:

$$\{A((h_1^{d-1}(k), h_2^{d-1}(k), \dots, \\ h_{d-1}^{d-1}(k)), h_d^1(j))\}_{k=1, j=1}^{k=r, j=r} \tag{7}$$

r combinations of $\{(h_1^{d-1}(k), h_2^{d-1}(k), \dots, h_{d-1}^{d-1}(k)\}_{k=1}^{k=r}$ hyperparameters are fixed.

The end of one "sweep right" is reached.

7. Similar to step 3, we have r combinations of hyperparameters, but they are not random anymore. We next perform for a similar procedure in the reverse direction (from the last hyperparameter to the first). The process is called the "sweep left."
   One first changes the index order:

$$\{(h_1^{d-1}(k), h_2^{d-1}(k), \dots, h_{d-1}^{d-1}(k)\}_{k=1}^{k=r} \Longrightarrow \text{relabel}$$

$$\{(h_d^{d-1}(k), h_{d-2}^{d-1}(k), \dots, h_2^{d-1}(k)\}_{j=1}^{j=r} \tag{8}$$

And then, continues from the fourth step of the TetraOpt algorithm.

8. A combination of the "sweep right" and the "sweep left" is a full sweep. We do n_{swp} full sweeps in this algorithm.
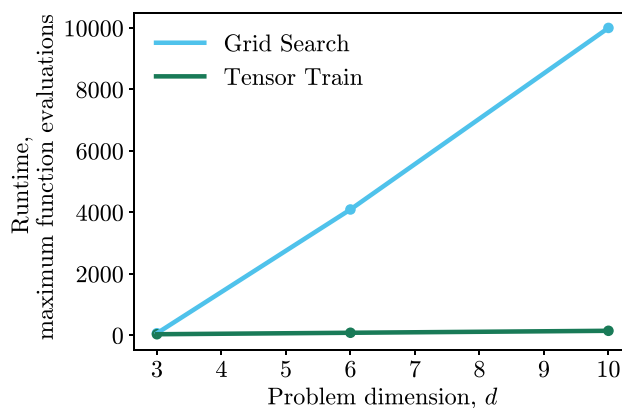

Fig. 2 Tensor train (TT) and grid search (GS): expected runtime in maximum objective function evaluations vs. growth of problem dimension d

9. During all the iterations, we record it if we estimate a new maximum score. An expected runtime comparison of this method against grid search for increasing problem dimensionality is shown in Fig. 2.

2.3 Benchmarking HPO methods

In order to ascertain the solution quality in our proposed method for hyperparameter optimization, we tested over three black-box objective functions. These functions included the Schwefel, Fletcher-Powell, and Vincent functions from the `optproblems` Python library optproblems (2022). We ran 100 randomly initialized trails and recorded average fitness and maximum number of function evaluations in response to the change in the problem size d for each objective function. We compared grid search (GS) and tensor train (TT)—both tabular methods (Table 1) for hyperparameter optimization. For tensor train and grid search, we partitioned the hyperparameter ranges with 4 discrete points per hyperparameter. For tensor train, we set the rank parameter $r = 2$.

2.4 Car classification with hybrid quantum neural networks

Computer vision and classification systems are ubiquitous within the mobility and automotive industries. In this article, we investigate the car classification problem using the car dataset Krause et al. (2013) provided by Stanford CS Department. Examples of cars in the data set are shown in Fig. 3. The Stanford Cars data set contains 16,185 images of 196 classes of cars. The data is split into 8144 training images and 8041 testing images. The classes are typically at the combination of make, model, year, e.g., Volkswagen Golf Hatchback 1991 or Volkswagen Beetle Hatchback 2012. Since the images in this data set have different sizes, we resized all images to

Table 1 Table of results comparing HPO methods for Schwefel, Fletcher-Powell, and Vincent objective functions

| HPO method	Average fitness	d	ER
Schwefel			
TT	**−541.76**	**3**	**32**
GS	−541.76	3	64
TT	**−1083.53**	**6**	**80**
GS	−1083.53	6	4092
TT	**−1805.89**	**10**	**144**
GS	−1805.89	10	10,000
Fletcher-Powell			
TT	5136.64	3	32
GS	**4113.78**	**3**	**64**
TT	23,954.5	6	80
GS	**14,295.2**	**6**	**4092**
TT	78,101.4	10	144
GS	**36,890.11**	**10**	**10,000**
Vincent			
TT	−0.232	3	32
GS	**−0.243**	**3**	**64**
TT	−0.242	6	80
GS	**−0.243**	**6**	**4092**
TT	−0.241	10	144
GS	**−0.243**	**10**	**10,000**

Average fitness values and expected runtimes (ER) in maximum function evaluations were calculated over 100 runs for varying sizes of problem dimension d (lower is better). Methods obtaining the best average fitness are highlighted in bold, with ties broken by lower ER

400 by 400 pixels. In addition, we apply random rotations by maximum 15°, random horizontal flips, and normalization to the training data. For testing data, only normalization has been applied.

We use transfer learning to solve the car classification problem. Transfer learning is a powerful method for training neural networks in which experience in solving one problem helps in solving another problem Neyshabur et al. (2020). In our case, the ResNet (residual neural network) He et al. (2015) is pretrained on the ImageNet dataset Imagenet dataset (2022) and is used as a base model. One can fix the weights of the base model, but if the base model is not flexible enough, one can "unfreeze" certain layers and make it trainable. Training deep networks is challenging due to the vanishing gradient problem, but ResNet solves this problem with so-called residual blocks: inputs are passed to the next layer in the residual block. In this way, deeper layers can see information about the input data. ResNet has established itself as a robust network architecture for solving image classification problems. We dowloaded ResNet34 via PyTorch PyTorch (2022), where the number after the model name, 34, indicates the number of layers in the network.

As shown in the Fig. 3(a), in the classical network after ResNet34, we add three fully connected layers. Each output neuron corresponds to a particular class of the classification problem, e.g., Volkswagen Golf Hatchback 1991 or Volkswagen Beetle Hatchback 2012. The output neuron with the largest value determines the output class. Since the output from the ResNet34 is composed of 512 features, the first fully connected layer consists of 512 input neurons and a bias neuron and n output features. The second fully connected layer connects n input neurons and a bias neuron with nq output features. The value of n and q can vary, thus changing the number of weights in the classical network. Since the network classifies k classes in the general case, the third fully connected layer takes nq neurons and a bias neuron as input and feeds k neurons as output.

In the hybrid analog as shown in Fig. 3(b), we replace the second fully connected layer with a quantum one. It is
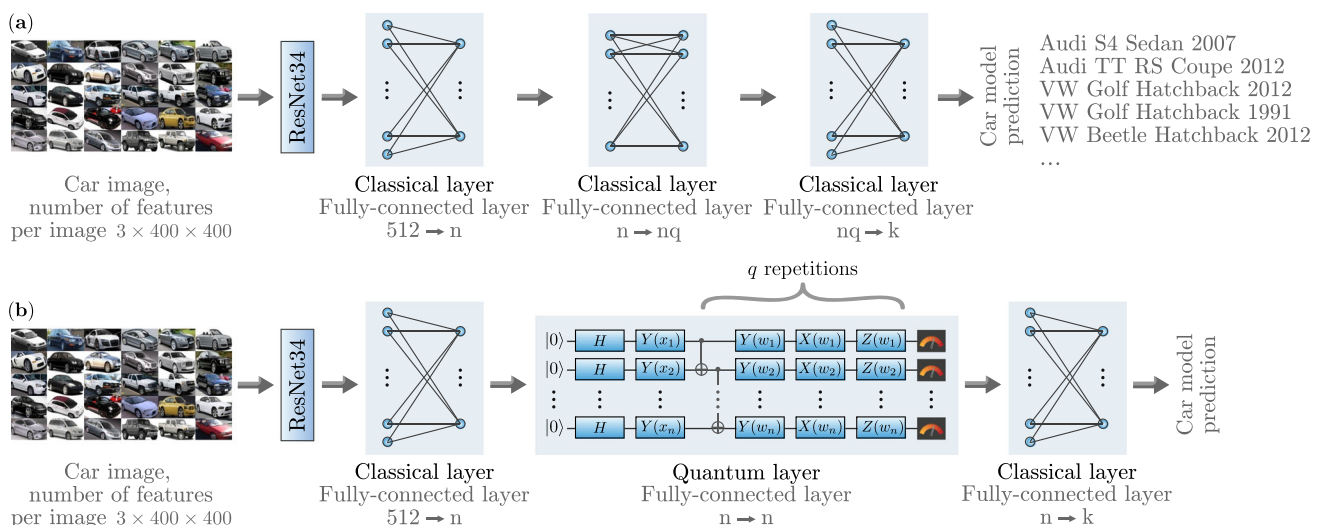
Fig. 3 Classical (a) and hybrid quantum (b) ResNet architectures

worth noting that the number of qubits used for the efficient operation of the model is initially unknown. The position of this layer was chosen to be between two classical layers that can appropriately pre-process the outputs of the ResNet (the first classical layer) and to post-process the quantum outputs (the final classical layer). In the quantum layer, the Hadamard transform is applied to each qubit, then the input data is encoded into the angles of rotation along the y-axis. The variational layer consists of the application of the CNOT gate and rotation along x, y, and z-axes. The number of variational layers can vary. Accordingly, the number of weights in the hybrid network can also change. The measurement is made in the X-basis. For each qubit, the local expectation value of the X operator is measured. This produces a classical output vector, suitable for additional post-processing. Since the optimal number of variational layers (q, depth of quantum circuit) and the optimal number of qubits n are not known in advance, we choose these values as hyperparameters. A thorough analysis of the quantum circuit for $n = 2$ is given in the Appendix, where three approaches are employed to measure the efficiency, trainability, and the expressivity of this quantum model.

We use the cross-entropy as a loss function

$$l = -\sum_{c=1}^{k} y_c \log p_c \qquad (9)$$

where p_c is the prediction probability, y_c is 0 or 1, determining respectively if the image belongs to the prediction class, and k is the number of classes. We use the Adam optimizer Adam optimizer (2022); Kingma and Ba (2014) and reduce the learning rate after several epochs. Note that in the simulation of the HQNN we assumed a precise (infinite-shots) and noise-free simulator, as investigating the effects of these sources of noise fell outside the scope this work. There is no one-size-fits-all rule of how to choose a learning rate. Moreover, in most cases, dynamic control of the learning rate of a neural network can significantly improve the efficiency of the backpropagation algorithm. For these reasons,

we choose the initial learning rate, the period of learning rate decay, and the multiplicative factor of the learning rate decay as hyperparameters. In total, together with number of variational layers and number of qubits, we optimize five hyperparameters presented in Table 2 to improve the accuracy of solving the problem of car classification.

2.5 Simulation results

We next perform a simulation of the hybrid quantum ResNet described in the previous section. The simulation is compared to its classical analog, the residual neural network, in a test car classification task. Because of the limited number of qubits available and computational time constraints, we used a classification between two classes, Volkswagen Golf Hatchback 1991 and Volkswagen Beetle Hatchback 2012, to compare the classical and hybrid networks fairly. In total, we used 88 testing images and 89 training images. Both the HQNN model and the classical NN model were used together with the GS and TetraOpt methods for hyperparameter optimization. All machine learning simulations were carried out in the QMware cloud, on which the classical part was implemented with the PyTorch framework, and the quantum part was implemented with the <basiq> SDK QMware (2022); Perelshtein et al. (2022); Kordzanganeh et al. (2022). The results of the simulations are shown in Fig. 4.

Figure 4(a) shows the dependence of accuracy on the number of HPO iterations on the test data, where one iteration of HPO is one run of the model. Green color shows the dependence of accuracy on the number of iterations for the HQNN, and blue color shows for the classical NN. As one can see from Fig. 4(a), TetraOpt works more efficiently than GS and in fewer iterations finds hyperparameters that give an accuracy above 0.9. HQNN with TetraOpt (marked with green crosses) finds a set of hyperparameters that yields 97.7% accuracy over 18 iterations. As for the GS (marked solid green line), it took 44 iterations to pass the threshold of 98% accuracy.

Table 2 The table shows which hyperparameters are being optimized, their labels, limits of change, and the best values found during HPO

Hyperparameter	Label	Range	Hybrid HPO values	Classical HPO values
Number of qubits number of neurons	n	$4 - 16$	13	5
Depth of quantum circuit	q	$1 - 5$	4	$\times$
Number of neurons	nq	$4 - 80$	$\times$	80
Initial learning rate	α_0	$1 - 10 \times 10^{-4}$	5×10^{-4}	5×10^{-4}
Step of learning rate	α_δ	$1 - 8$	8	5
Multiplicative factor of earning rate decay	α_r	$0.1 - 0.2$	0.1	0.2
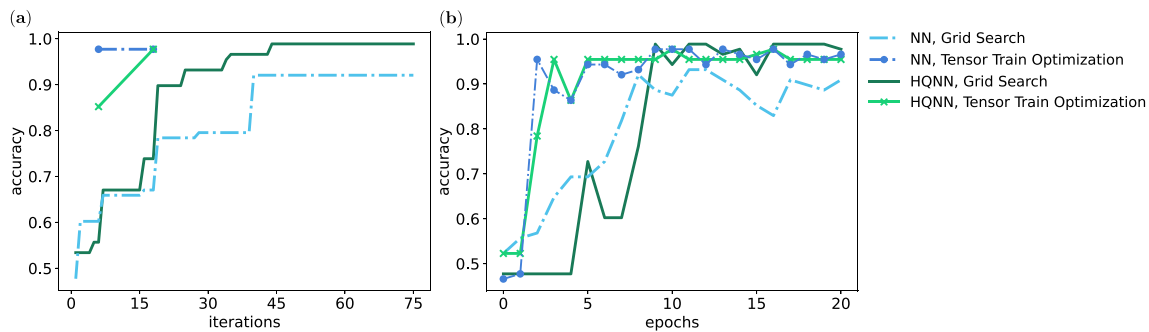
Fig. 4 (a) Dependence of accuracy on the number of iterations HPO. TetraOpt for the hybrid model found a set of hyperparameters that gives an accuracy of 0.852 after 6 iterations, 0.977 after 18 iterations, for the classical model found 0.977 after 6 iterations. Grid search for the hybrid model found a set of hyperparameters that gives an accuracy of 0.989 after 75 iterations, for the classical model found 0.920 after 75 iterations. (b) Dependence of accuracy on the number of epochs with the found optimal set of hyperparameters

TetraOpt finds in 6 iterations a set of hyperparameters for the classical NN, which gives an accuracy of 97.7%, which is the same as the accuracy given by the set of hyperparameters for the HQNN that found in 18 iterations. As for the GS, it is clear that the optimization for the HQNN works more efficiently than for the classical one. And the optimization of the HQNN requires fewer iterations to achieve higher accuracy compared to the optimization of the classical NN. A possible reason is that a quantum layer with a relatively large number of qubits and a greater depth works better than its classical counterpart.

The best values found during HPO are displayed in Table 2. The quantum circuit corresponding to the optimal set of hyperparameters has 52 variational parameters, leading to a total of 6749 weights in the HQNN. In the classical NN, there are 9730 weights. Therefore, there are significantly fewer weights in a HQNN compared to a classical NN. Nevertheless, as can be seen from the Fig. 4(b), the HQNN,

with the hyperparameters found using the GS, reaches the highest overall accuracy (98.9%). Figure 5 shows examples of car images that were classified correctly by the HQNN model. The HQNN with an optimized set of hyperparameters achieved an accuracy of 0.989.

3 Discussion

We introduced two new ML developments to image recognition. First, we presented a quantum-inspired method of tensor train decomposition for choosing ML model hyperparameters. This decomposition enabled us to optimize hyperparameters similar to other tabular search methods, e.g., grid search, but required only $\mathcal{O}(dnr^2)$ hyperparameter choices instead of $\mathcal{O}(n^d)$ in the grid search method. We verified this method over various black-box functions and found that the tensor train method achieved comparable results in average fitness, with a reduced expected run time for most of the test functions compared to grid search. This indicates that this method may be useful for high dimensional hyperparameter searches for expensive black-box functions. Future work could investigate using this method in combination with local search heuristic, where the tensor train optimizer performs a sweep over a larger search space within a budget and seeds another optimization routine for a local search around this region. This method could also be applied to the B/n problem for successive halving algorithm by decomposing the search space to find the optimal ratio of budget B over configurations n. Future work could investigate these applications in more detail.

Second, we presented a hybrid quantum ResNet model for supervised learning. The hybrid model consisted of the combination of ResNet34 and a quantum circuit part, whose size and depth became the hyperparameters. The size and flexibility of the hybrid ML model allowed us to apply it to car image classification. The hybrid ML model with GS showed an accuracy of 0.989 after 75 iterations in our binary clas-

Volkswagen Golf Hatchback 1991

Volkswagen Beetle Hatchback 2012
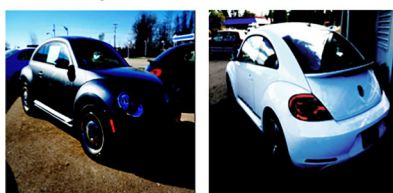
Fig. 5 Examples of test car images that were correctly classified by the hybrid quantum ResNet

sification tests with images of Volkswagen Golf Hatchback 1991 and Volkswagen Beetle Hatchback 2012. This accuracy was better than of a comparable classical ML model with GS showed an accuracy of 0.920 after 75 iterations. In the same test, the hybrid ML model with TetraOpt showed an accuracy of 0.977 after 18 iterations, whereas the comparable classical ML model with TetraOpt showed the same accuracy of 0.977 after 6 iterations. Our developments provide new ways of using quantum and quantum-inspired methods in practical industry problems. In future research, exploring the sample complexity of the hybrid quantum model is of importance, in addition to generalization bounds of the quantum models similar to research in Caro et al. (2022). Future work could also entail investigating state-of-the-art improvements in hyperparameter optimization for classical and quantum-hybrid neural networks and other machine learning models by leveraging quantum-inspired or quantum-enhanced methods.

Appendix A: Quantum Circuit Analysis

In this section, we critically analyze the parameterized quantum circuit (PQC) suggested in Section 2.4.

There are many methods to do this and in this paper we focus on three of them:

- ZX calculus circuit-reducibility as suggested in Coecke and Duncan (2008)
- Fisher information degeneracy and the effective dimension as suggested in Abbas et al. (2021)
- Fourier accessibility, first suggested in Schuld et al. (2021)

We see that the circuit in use is optimally chosen based on these measures.

Appendix A.1: ZX calculus

ZX calculus is a graphical language that can reduce a circuit to an identical, see Coecke and Duncan (2008). To reduce a circuit using ZX calculus we need to first convert the quantum circuit to a ZX graph. Then we can use the ZX calculus rules, suggested in van de Wetering (2020), to reduce this graph to a more fundamental version of itself. We then convert the reduced ZX graph back to a new and reduced circuit. If a circuit cannot be reduced, we shall refer to it as ZX-irreducible. A circuit of this type can use the maximum potential of the trainable layers and includes no *fully redundant parameters*. Our circuit produces the graph in Fig. 6. None of the parameterized gates shown in this figure can commute or be simplified, and therefore our circuit is ZX-irreducible. Specifically, the following two crucial steps were taken to make sure that this is the case:

- Due to the final R_Z rotation gates, measurements were made in the X-basis to make sure these gates were not made redundant, and
- R_Y rotation gates were employed to prevent the non-commutativity through the CNOT gates.

Although ZX-irreducibility is a crucial condition to look for, further analysis is required to understand the expressivity and the efficiency of the circuit.

Appendix A.2: Fisher information and effective dimension

We can summarize supervised machine learning as creating a hypothesis model $h_{\theta}(\hat{x})$ from a labelled dataset $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ that could produce an approximation to the distribution of the data in nature, $f(\mathbf{x})$. We are provided with a subset of S labelled data points from this distribution and we need to optimize our hypothesis model to be a representative model of $f(\hat{x})$.

To do this, we need to maximize the probability that given the model parameters $\boldsymbol{\theta}$ and some data point $\mathbf{x}$ we get the associated label y. This conditional probability can be written as $P(y|\mathbf{x}, \boldsymbol{\theta})$. However, the latter notion assumes a uniform distribution over $\mathcal{X}$, and to be more accurate we need to consider the joint probability, $P(y, \mathbf{x}|\boldsymbol{\theta})$. The joint probability distribution can be empirically evaluated for any value of θ for a given subset of data. Thus, we can think of the joint probability as an N-dimensional manifold where N is the number
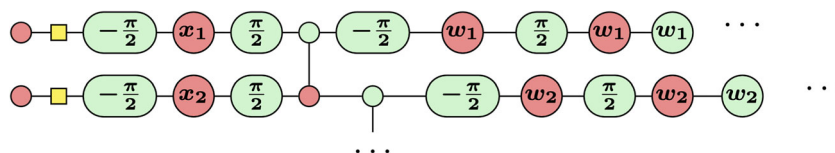
Fig. 6 The only changes we could make to this circuit are fusing some constant spiders, which we will need to re-introduce later for circuit efficiency. Additionally, measurements are in the X-basis, so all variational parameters to the right of the last CNOT only contribute to the qubit that they are applied to. This is particularly evident in Fig. 10(a) and (e), where there is only one CNOT in the system. This allows us to assign a variable specific to each qubit which we can use to tune the output of each qubit independently

of trainable parameters $N = |\boldsymbol{\theta}|$. The Fisher information matrix $F(\boldsymbol{\theta})$ can define a metric over this manifold Abbas et al. (2021); Amari (1998)

$$F(\boldsymbol{\theta}) = \mathbb{E}_{\{x_i, y_i\}}[\nabla_{\boldsymbol{\theta}} \log(P) \nabla_{\boldsymbol{\theta}} \log(P)^T]. \qquad (10)$$

This metric can be diagonalized to produce a locally Euclidean tangential basis whose diagonal values provide the square of the gradient of our joint probability in this diagonalized basis. These values can be obtained by calculating the eigenvalues of the Fisher matrix. To understand the usefulness of this insight, we need to understand the issue of barren plateaus in quantum neural networks (QNNs). McClean et al. (2018) suggested that for a chosen QNN, the expectation values of the gradients are zero and their variances decrease exponentially with the number of qubits. This combination means that QNNs suffer from vanishing gradients, a phenomenon known as the barren plateau problem. We must avoid these barren plateaus by ensuring our network can produce a spectrum of gradients rather than a large number of zeros. We showed that the eigenvalues of the Fisher information matrix produced the square of our gradients. Therefore, by calculating the eigenvalue spectrum of Fisher matrices for many realizations of $\boldsymbol{\theta}$ we can investigate the trainability - the robustness of the QNN against barren plateaus - of the specific 2-qubit network. It is noteworthy that the barren plateau phenomenon scales exponentially with the qubit count and that this section of the analysis is only applicable to the 2-qubit case. A network with high trainability would have a lower eigenvalue degeneracy about zero.

Berezniuk et al. (2020) takes this concept a step further by assuming that - under some weak conditions - the Fisher information matrix above is equal[1] to the Hessian matrix defined as

$$H(\boldsymbol{\theta}) = \mathbb{E}_{\{x_i, y_i\}}[\nabla^T \nabla \log(P)], \qquad (11)$$

which is the matrix of second-order derivatives. Then, it uses this equivalence to derive a complexity measure that is dependent on the size of the subset S. This measure of complexity is defined as the effective dimension and was first practically explored in Abbas et al. (2021) to show that QNNs can have a higher expressivity than classical machine learning models. The latter work defines the effective dimension as

$$d_{\gamma, S}(\mathcal{M}_{\Theta}) := 2 \frac{\log\left(\frac{1}{V_{\Theta}} \int_{\Theta} \sqrt{\det\left(i d_N + \frac{\gamma S}{2\pi \log S} \hat{F}(\boldsymbol{\theta})\right)} d\boldsymbol{\theta}\right)}{\log\left(\frac{\gamma S}{2\pi \log S}\right)}, \qquad (12)$$

---

[1] It is noteworthy that this idea is sometimes contested in the statistical learning literature Thomas et al. (2020); Kunstner et al. (2020).

where $V_{\Theta} := \int_{\Theta} d\theta$ is the volume of the parameter space, γ is a constant in $(0, 1]$ introduced in Abbas et al. (2021), and $\hat{F}(\boldsymbol{\theta})$ is the normalized Fisher matrix defined as

$$\hat{F}_{ij}(\boldsymbol{\theta}) := N \frac{V_{\Theta}}{\int_{\Theta} tr(F(\boldsymbol{\theta})) d\boldsymbol{\theta}} F_{ij}(\boldsymbol{\theta}). \qquad (13)$$

We can calculate the Fisher information for the specific hyperparameter settings of our circuit. Specifically, we consider a 2-qubit variation of this circuit with the number of trainable layers varying from 1 to 20. Following the lead of Abbas et al. (2021), we create a Gaussian dataset $\mathbf{x} \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$ and evaluate the joint probability by overlapping the specific resultant state with the state of our QNN

$$P(y, \mathbf{x}|\boldsymbol{\theta}) = |\langle y \mid \psi(\boldsymbol{\theta}, \mathbf{x}) \rangle|^2, \qquad (14)$$

where y is the output state. Note that this has to be averaged over all possible y and $\mathbf{x}$. This way, we can calculate the empirical Fisher information for any $\boldsymbol{\theta}$. Figure 7 shows the mean-square, normalised Fisher matrix for 1000 data points and 100 uniform weight realizations $\theta \in (0, 2\pi]$. Observing the diagonal elements, it seems that none of the parameters is especially dominant or redundant. A further test would be to look at the Fisher eigenvalue spectra shown in Fig. 8. We can see that the degeneracy of the eigenvalues around zero increases for a higher number of trainable layers.

Finally, to obtain the effective dimension, we can evaluate the integral in Eq. 12 by taking the average of 100 Fisher realizations. Figure 9(a) shows the effective dimension against the number of trainable layers of our network. Increasing the number of trainable layers increases the effective dimension. This is unsurprising as we defined the effective dimension as a measure of expressivity and we expect that adding trainable layers would increase the expressivity of the network. However, we also see that adding trainable layers could yield diminishing returns at higher values.

Additionally, it was shown in Larocca et al. (2021) that certain QNNs can become over-parameterized and exhibit lowered parameter efficiency. This was quantified by finding the parameterization for which, at least at one point in the loss landscape, any added parameter would leave the rank of the Fisher information matrix unchanged - in other words, the rank of the Fisher matrix becomes saturated for an over-parameterized circuit. Examining Fig. 9(b), we see the FIM rank of the circuit increases linearly with the number of trainable parameters and then plateaus at 6 trainable layers, reaching a maximal rank of $r = 12$. This means that although the effective dimension seems to increase beyond this point, but the circuit is saturated and there is no further increase in expressivity.

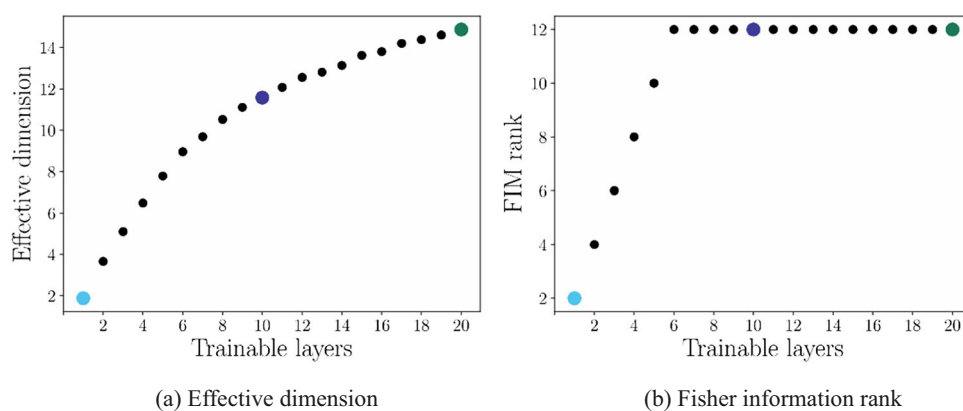(a) 1 trainable layer　　　　　(b) 10 trainable layers　　　　　(c) 20 trainable layers
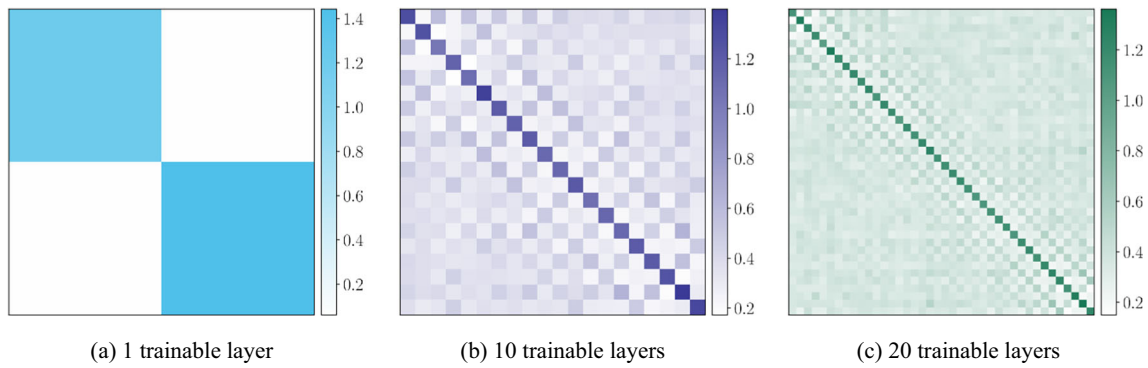
Fig. 7 The square-averaged normalized Fisher matrices. The diagonals of these matrices show that the quantum circuit distributes the gradients to all trainable parameters and there is no evident single-parameter dominance
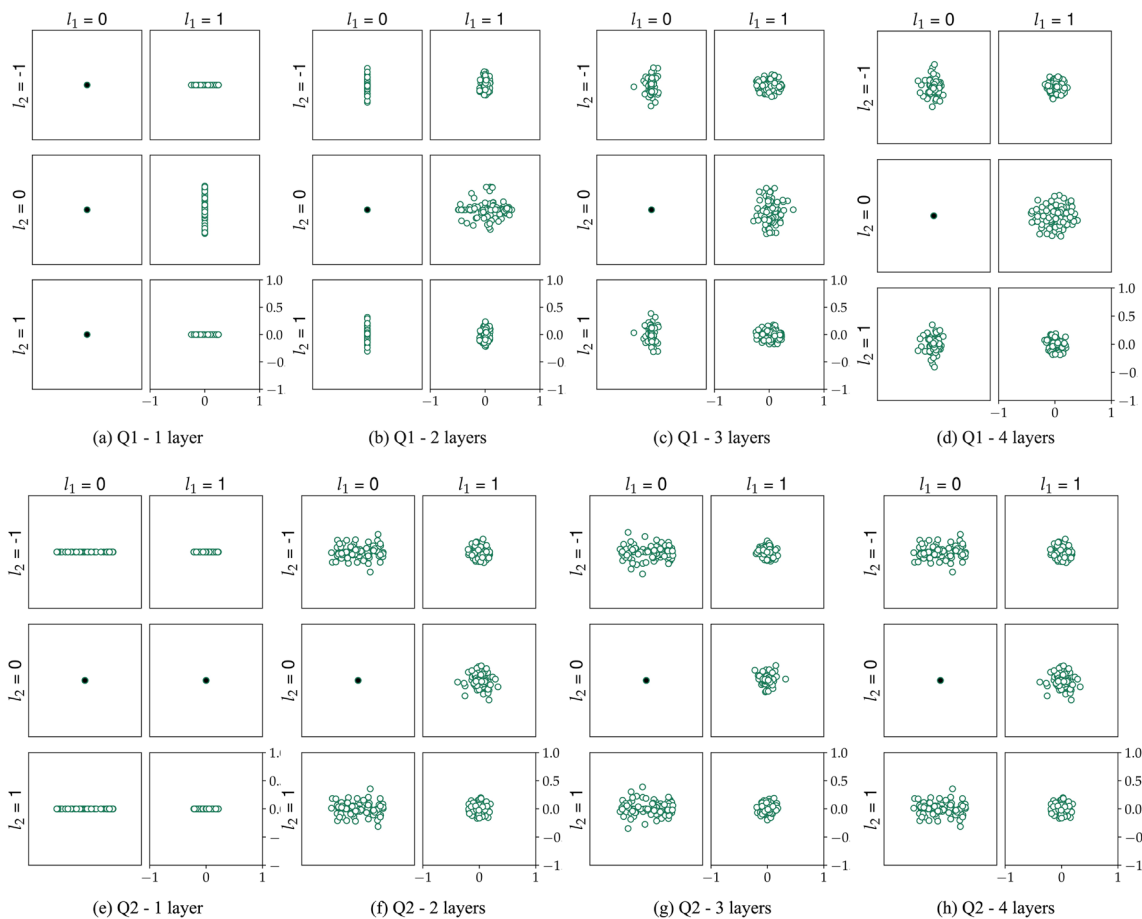

(a) 1 trainable layer　　　　　(b) 10 trainable layers　　　　　(c) 20 trainable layers
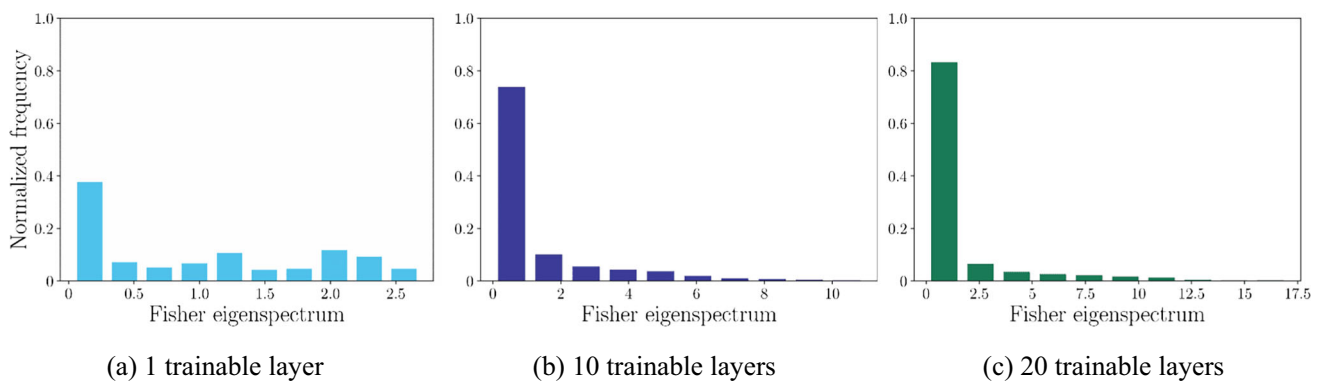
Fig. 8 The normalized histogram of the Fisher eigenvalue spectra. Degeneracy about zero means that a smaller portion of the parameters is effectively driving the training, i.e. a lower trainability. (a) has the highest trainability and (c) the lowest. This means that by increasing the number of trainable layers we decrease the trainability of the model. We see in Fig. 9 that this decrease in trainability is accompanied by an increase in model expressivity, presenting a trade-off for model selection

(a) Effective dimension　　　　　　　(b) Fisher information rank

Fig. 9 (a) shows that the effective dimension increases with the number of trainable layers, and (b) illustrates the limit of over-parameterization of the circuit. The colored points light blue, navy, and green correspond to the circuits with 1, 10, and 20 trainable layers, respectively

(a) Q1 - 1 layer   (b) Q1 - 2 layers   (c) Q1 - 3 layers   (d) Q1 - 4 layers

(e) Q2 - 1 layer   (f) Q2 - 2 layers   (g) Q2 - 3 layers   (h) Q2 - 4 layers

Fig. 10 The Fourier accessibility of our circuit when employed for two features. Note that l_1 is only shown to span the positive half of its indices and this is due to the symmetry of these coefficients $c_{l_1,l_2} = c_{-l_1,-l_2}$. In all figures, we see that the circuit is not able to change the offset coefficient. This sets a constraint but also increases our accessibility because at a maximum the amplitude of the Fourier output cannot exceed 1 (as the expectation value of our circuit needs to remain between -1 and 1). Figures (a) and (e) respectively show the output of the first and the sec-

ond qubits when only one trainable layer is implemented. In this case, the associated qubit has an asymmetric advantage in accessibility. We also see that in this case, the phases of our coefficients remain fixed. Both of the mentioned issues can be improved by adding trainable layers, and Figs. (d) and (h) corroborate this statement. Note that this does not show the full extent of Fourier accessibility as a complete investigation would also look into the inter-dependence of these coefficients, but that lies outside the scope of this work

These analyses signify a trade-off between trainability, determined by the eigenvalue spectra in Fig. 8, and the expressivity quantified by the effective dimension and upperbound by the maximal rank.

Appendix A.3: Fourier accessibility

Schuld et al. (2021) showed that a QNN that uses angle-embedding[2] produces a truncated Fourier series of degree L. This degree is dependent on the number of encoding repetitions employed in a QNN, a strategy first employed in Pérez-Salinas et al. (2020). Furthermore, Schuld et al. (2021);

Kordzanganeh et al. (2021) showed that for a multi-feature setting we get a multi-dimensional truncated Fourier series. For a two-feature setting, we get

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \langle \psi(\boldsymbol{\theta}, \boldsymbol{x}) | M | \psi(\boldsymbol{\theta}, \boldsymbol{x}) \rangle \tag{15}$$

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \sum_{l_1=-L_1}^{L_1} \sum_{l_2=-L_2}^{L_2} 2|c_{l_1,l_2}| cos(l_1 x_1 + l_2 x_2 - arg(c_{l_1,l_2})), \tag{16}$$

where $|\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle$ is the quantum state of the system after encoding and variational layers, M is the measurement gate, and L_1 and L_2 are the number of encoding repetitions of the first and the second feature respectively. The complex coefficients c_{l_1,l_2} determine the amplitude and the phase of

---

[2] Schuld (2021) provided expressivity analysis for different angle embedding strategies.

each Fourier term. These coefficient depend only on the variational gates, and so, our accessibility to a full Fourier series is limited by how these variational gates span the Fourier space. We can investigate a specific subset of our networks with 2 features and a single encoding repetition. This means that our circuit has $L_1 = L_2 = 1$. Thus, we can set up the circuit and randomly realize the weights many times to assess the Fourier accessibility of the circuit. Figure 10 shows the Fourier accessibility of our network for 100 uniform realizations of weights $\theta \in [0, 2\pi)^N$. It is evident that increasing the number of trainable layers improves the Fourier accessibility of the QNN. Furthermore, we can see that to have an unimpeded network we need at least 3 layers of variational gates.

Appendix A.4: Summary

In this analysis, we assessed the feasibility of the chosen quantum circuit and looked at three approaches for analyzing its effectiveness: ZX-reducibility, Fisher information, and Fourier analysis. In Appendix A.1 we proved that there are no redundant parameters in the circuit caused by commutation of the quantum gates and that certain weights are reserved for independent contribution to each qubit. Then, in Appendix A.2 we showed that none of the parameters dominated the training and that by increasing the number of trainable layers, the trainability and the complexity of our model respectively decreased and increased. The increase in model complexity stopped at 6 layers, where we showed that the rank of the Fisher information matrix was saturated and any additional parameterization would be futile. Finally, in Appendix A.3 we used the theoretical findings of Schuld et al. (2021) to show that for a 2-qubit version of our network we at least needed 3 layers of variational gates to represent the full Fourier landscape.

Declarations

References

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

Adam optimizer. https://pytorch.org/docs/stable/generated/torch.optim.Adam.html (2022)

Alcazar J, Leyton-Ortega V, Perdomo-Ortiz A (2020) Classical versus quantum models in machine learning: insights from a finance application. Mach Learn: Sci Technol 1(3):035003. https://doi.org/10.1088/2632-2153/ab9009

Amari S-i (1998) Natural gradient works efficiently in learning. Neural Computat 10(2):251–276. https://doi.org/10.1162/089976698300017746

Amaro D, Rosenkranz M, Fitzpatrick N, Hirano K, Fiorentini M (2022) A case study of variational quantum algorithms for a job shop scheduling problem. EPJ Quantum Technol 9:5. https://doi.org/10.1140/epjqt/s40507-022-00123-4

Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, Biswas R, Boixo S, Brandao FGSL, Buell DA et al (2019) Quantum supremacy using a programmable superconducting processor. Nature 574(7779):505–510. https://doi.org/10.1038/s41586-019-1666-5

Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, Boixo S, Broughton M, Buckley BB, Buell DA et al (2020) Hartree-Fock on a superconducting qubit quantum computer. Science 369(6507):1084–1089. https://doi.org/10.1126/science.abb9811

Awad N, Shala G, Deng D, Mallik N, Feurer M, Eggensperger K, Biedenkapp A, Vermetten D, Wang H, Doerr C, Lindauer M, Hutter F (2020) Squirrel: a switching hyperparameter optimizer. arXiv:2012.08180

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming. Oxford University Press Inc, USA, Genetic Algorithms. https://doi.org/10.1093/oso/9780195099713.001.0001

Berezniuk O, Figalli A, Ghigliazza R, Musaelian K (2020) A scale-dependent notion of effective dimension. arXiv:2001.10872

Caro MC, Huang H-Y, Cerezo M, Sharma K, Sornborger A, Cincio L, Coles PJ (2022) Generalization in quantum machine learning from few training data. Nat Commun 13:4919. https://doi.org/10.1038/s41467-022-32550-3

Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, McClean JR, Mitarai K, Yuan X, Cincio L, et al. (2021) Variational quantum algorithms. Nature Rev Phys 3(9), 625–644. https://doi.org/10.1016/j.laa.2009.07.024

Cirac JI, Pérez-García D, Schuch N, Verstraete F (2021) Matrix product states and projected entangled pair states: concepts, symmetries,

theorems. Rev Modern Phys 93(4): 045003. https://doi.org/10.1038/s42254-021-00348-9

Coecke B, Duncan R (2008) Interacting quantum observables. International Colloquium on Automata, Languages, and Programming, pages 298–310, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-70583-3_25

Dalyac C, Henriet L, Jeandel E, Lechner W, Perdrix S, Porcheron M, Veshchezerova M (2021) Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles. EPJ Quantum Technol 8(1):12. https://doi.org/10.1140/epjqt/s40507-021-00100-3

Dou T, Wang K, Zhou Z, Shilu Yan, and Wei Cui (2021) An unsupervised feature learning for quantum classical convolutional network with applications to fault detection. In 2021 40th Chinese Control Conference (CCC), pages 6351–6355. IEEE. https://doi.org/10.23919/ccc52363.2021.9549885

Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(1):1997–2017. https://doi.org/10.1109/access.2021.3126685

Feld S, Roch C, Gabor T, Seidel C, Neukart F, Galter I, Mauerer W, Linnhoff-Popien C (2019) A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. Front Inf Commun Technol 6:13. https://doi.org/10.3389/fict.2019.00013

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown (2011) Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, Learning and Intelligent Optimization, pages 507–523. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25566-3_40

Gómez RB, O'Meara C, Cortiana G, Mendl CB, Bernabé-Moreno J (2022) Towards AutoQML: a cloud-based automated circuit architecture search framework. arXiv:2202.08024

Goreinov S, Oseledets I, Savostyanov D, Tyrtyshnikov E, Zamarashkin N (2010) How to find a good submatrix. Theory, Algorithms Appl, Matrix Methods. https://doi.org/10.1142/9789812836021_0015

Grant E, Humble TS, Stump B (2021) Benchmarking quantum annealing controls with portfolio optimization. Phys Rev Appl 15(1):014012. https://doi.org/10.1103/physrevapplied.15.014012

Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum enhanced feature spaces. Nature 567(7747):209–212. https://doi.org/10.1038/s41586-019-0980-2

He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385

Hutter F, Kotthoff L, Vanschoren J (2019) Automated machine learning: methods, systems. Challenges, Springer Nature. https://doi.org/10.1007/978-3-030-05318-5

ImageNet dataset. https://image-net.org/ (2022)

Johnson MW, Amin MHS, Gildert S, Lanting T, Hamze F, Dickson N, Harris R, Berkley AJ, Johansson J, Bunyk P et al (2011) Quantum annealing with manufactured spins. Nature 473(7346):194–198. https://doi.org/10.7566/jpsj.91.044003

Kabanikhin S, Krivorotko O, Zhang S, Kashtanova V, Wang Y (2019) Tensor train optimization for mathematical model of social networks. arXiv:1906.05246

Karnin Z, Koren T, Somekh O (2013) Almost optimal exploration in multi-armed bandits. In Sanjoy Dasgupta and David McAllester, editors, Proceedings of the 30th International Conference on Machine Learning, volume 28 of Proceedings of Machine Learning Research, pages 1238–1246, Atlanta, Georgia, USA. PMLR. https://proceedings.mlr.press/v28/karnin13.html

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980

Kohavi R, John GH (1995) Automatic parameter selection by minimizing estimated error. In: Armand Prieditis, Stuart Russell (eds) Machine Learning Proceedings 1995. Morgan Kaufmann,

San Francisco (CA), pp 304–312. https://doi.org/10.1016/b978-1-55860-377-6.50045-1

Kordzanganeh M, Buchberger M, Kyriacou B, Povolotskii M, Fischer W, Kurkin A, Somogyi W, Sagingalieva A, Pflitsch M, Melnikov A (2023) Benchmarking simulated and physical quantum processing units using quantum and hybrid algorithms. Adv Quantum Technol 6(8):2300043. https://doi.org/10.1002/qute.202300043

Kordzanganeh M, Sekatski P, Fedichkin L, Melnikov A (2023) An exponentially-growing family of universal quantum circuits. Mach Learn: Sci Technol 4(3):035036. https://doi.org/10.1088/2632-2153/ace757

Kordzanganeh M, Utting A, Scaife A (2021) Quantum machine learning for radio astronomy. arXiv:2112.02655

Krause J, Stark M, Deng J, Fei-Fei L (2013) 3D object representations for fine-grained categorization. In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia. https://doi.org/10.1109/iccvw.2013.77

Kunstner F, Balles L, Hennig P (2020) Limitations of the empirical Fisher approximation for natural gradient descent. arXiv:1905.12558

Larocca M, Ju N, García-Martín D, Coles PJ, Cerezo M (2023) Theory of overparametrization in quantum neural networks. Nat Comput Sci 3(6):542–551. https://doi.org/10.1038/s43588-023-00467-6

Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2017) Hyperband: a novel bandit-based approach to hyperparameter optimization. J Mach Learn Res 18(1):6765–6816. https://dl.acm.org/doi/abs/10.5555/3122009.3242042

Li L, Jamieson KG, Rostamizadeh A, Gonina E, Hardt M, Recht B, Talwalkar A (2018) Massively parallel hyperparameter tuning. arXiv:1810.05934

Lindauer M and Hutter F (2018) Warmstarting of model-based algorithm configuration. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). https://doi.org/10.1609/aaai.v32i1.11532

Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. Nature Phys 17(9):1013–1017

Luckow A, Klepsch J, Pichlmeier J (2021) Quantum computing: towards industry reference problems. Digitale Welt 5:34–45. https://doi.org/10.1007/s42354-021-0335-7

Malone FD, Parrish RM, Welden AR, Fox T, Degroote M, Kyoseva E, Moll N, Santagati R, Streif M (2022) Towards the simulation of large scale protein-ligand interactions on NISQ-era quantum computers. Chem Sci 13:3094. https://doi.org/10.1039/D1SC05691C

Mari A, Bromley TR, Izaac J, Schuld M, Killoran N (2020) Transfer learning in hybrid classical-quantum neural networks. Quantum 4:340. https://doi.org/10.22331/q-2020-10-09-340

McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nature Commun 9(1):4812. https://doi.org/10.1038/s41467-018-07090-4

Mehta A, Muradi M, Woldetsadick S (2019) Quantum annealing based optimization of robotic movement in manufacturing. In International Workshop on Quantum Technology and Optimization Problems pages 136–144. Springer. https://doi.org/10.1007/978-3-030-14082-3_12

Melnikov A, Kordzanganeh M, Alodjants A, Lee RK (2023) Quantum machine learning: from physics to software engineering. Advances in Physics: X 8(1):2165452. https://doi.org/10.1080/23746149.2023.2165452

Neukart F, Compostella G, Seidel C, Dollen DV, Yarkoni S, Parney B (2017) Traffic flow optimization using a quantum annealer. Front Inf Commun Technol 4:29. https://doi.org/10.3389/fict.2017.00029

Neyshabur B, Sedghi H, Zhang C (2020) What is being transferred in transfer learning? arXiv:2008.11687

Ohzeki M, Miki A, Miyama MJ, Terabe M (2019) Control of automated guided vehicles without collision by quantum annealer and digital devices. Front Comput Sci 1:9. https://doi.org/10.3389/fcomp.2019.00009

optproblems. https://pypi.org/project/optproblems/ (2022)

Oseledets I, Tyrtyshnikov E (2010) TT-cross approximation for multidimensional arrays. Linear Algebra Appl 432(1):70–88. https://doi.org/10.1016/j.laa.2009.07.024

Perelshtein M, Sagingalieva A, Pinto K, Shete V, Pakhomchik A, Melnikov A, Neukart F, Gesek G, Melnikov A, Vinokur V (2022) Practical application-specific advantage through hybrid quantum computing. arXiv:2205.04858

Pérez-Salinas A, Cervera-Lierta A, GilFuster E, Latorre JI (2020) Data re-uploading for a universal quantum classifier. Quantum 4:226. https://doi.org/10.22331/q-2020-02-06-226

Peters E, Caldeira J, Ho A, Leichenauer S, Mohseni M, Neven H, Spentzouris P, Strain D, Perdue GN (2021) Machine learning of high dimensional data on a noisy quantum processor. arXiv:2101.09581

Pramanik S, Chandra MG, Sridhar CV, Kulkarni A, Sahoo P, Vishwa Chethan DV, Sharma H, Paliwal A, Navelkar V, Poojary S, et al. (2021) A quantum-classical hybrid method for image classification and segmentation. arXiv:2109.14431

PyTorch. https://pytorch.org/ (2022)

QMware, The first global quantum cloud. https://qm-ware.com (2022)

Rainjonneau S, Tokarev I, Iudin S, Rayaprolu S, Pinto K, Lemtiuzhnikova D, Koblan M, Barashov E, Kordzanganeh M, Pflitsch M, Melnikov A (2023) Quantum algorithms applied to satellite mission planning for Earth observation. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 16:7062–7075. https://doi.org/10.1109/JSTARS.2023.3287154

Rebentrost P, Mohseni M, Lloyd S (2014) Quantum support vector machine for big data classification. Phys Rev Lett 113(13):130503. https://doi.org/10.1103/physrevlett.113.130503

Rudolph MS, Toussaint NB, Katabarwa A, Johri S, Peropadre B, Perdomo-Ortiz A (2022) Generation of high-resolution handwritten digits with an ion-trap quantum computer. Phys Rev X 12(3):031010. https://doi.org/10.1103/PhysRevX.12.031010

Sagingalieva A, Kordzanganeh M, Kenbayev N, Kosichkina D, Tomashuk T, Melnikov A (2023) Hybrid quantum neural network for drug response prediction. Cancers 15(10):2705. https://doi.org/10.3390/cancers15102705

Schuld M (2021) Supervised quantum machine learning models are kernel methods. arXiv:2101.11020

Schuld M, Bocharov A, Svore KM, Wiebe N (2020) Circuit-centric quantum classifiers. Phys Rev A 101(3):032308. https://doi.org/10.1103/physreva.101.032308

Schuld M, Killoran N (2019) Quantum machine learning in feature Hilbert spaces. Phys Rev Lett 122(4). https://doi.org/10.1103/physrevlett.122.040504

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). https://doi.org/10.1103/physreva.103.032430

Sebastianelli A, Zaidenberg DA, Spiller D, Saux BL, Ullo SL (2021) On circuit-based hybrid quantum neural networks for remote sensing imagery classification. IEEE J Selected Topics Appl Earth Observations Remote Sens 15:565–580. https://doi.org/10.1109/jstars.2021.3134785

Skolik A, McClean JR, Mohseni M, van der Smagt P, Leib M (2021) Layerwise learning for quantum neural networks. Quantum Mach Intell 3(1):1–11. https://doi.org/10.1007/s42484-020-00036-4

Skolik A, Jerbi S, Dunjko V (2022) Quantum agents in the gym: a variational quantum algorithm for deep Q-learning. Quantum 6:720. https://doi.org/10.22331/q-2022-05-24-720

Stollenwerk T, O'Gorman B, Venturelli D, Mandra S, Rodionova O, Ng H, Sridhar B, Rieffel EG, Biswas R (2019) Quantum annealing applied to deconflicting optimal trajectories for air traffic management. IEEE Trans Intell Transportat Syst 21(1):285–297. https://doi.org/10.1109/tits.2019.2891235

Streif M, Leib M (2020) Training the quantum approximate optimization algorithm without access to a quantum processing unit. Quantum Sci Technol 5(3):034008. https://doi.org/10.1088/2058-9565/ab8c2b

Streif M, Neukart F, Leib M (2019) Solving quantum chemistry problems with a D-Wave quantum annealer. In International Workshop on Quantum Technology and Optimization Problems, pages 111–122. Springer. https://doi.org/10.1007/978-3-030-14082-3_10

Streif M, Yarkoni S, Skolik A, Neukart F, Leib M (2021) Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. Phys Rev A 104(1): 012403. https://doi.org/10.1103/physreva.104.012403

Thomas V, Pedregosa F, Merriënboer BV, Mangazol P-A, Bengio Y, Roux NL (2020) On the interplay between noise and curvature and its effect on optimization and generalization. In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. http://proceedings.mlr.press/v108/thomas20a/thomas20a.pdf

van de Wetering J (2020) ZX-calculus for the working quantum computer scientist. arXiv:2012.13966

Vermetten D, Wang H, Doerr C, Bäck T (2020) Sequential vs. integrated algorithm selection and configuration: a case study for the modular cma-es. arXiv:1912.05899

Wang D, Zhao G, Chen H, Liu Z, Deng L, Li G (2021) Nonlinear tensor train format for deep neural network compression. Neural Netw 144:320–333. https://doi.org/10.1016/j.neunet.2021.08.028

White SR (1992) Density matrix formulation for quantum renormalization groups. Phys Rev Lett 69:2863–2866. https://doi.org/10.1103/physrevlett.69.2863

Xia R, Bian T, Kais S (2017) Electronic structure calculations and the Ising Hamiltonian. J Phys Chem B 122(13):3384–3395. https://doi.org/10.1021/acs.jpcb.7b10371

Yarkoni S, Alekseyenko A, Streif M, Dollen DV, Neukart F, Bäck T (2021) Multicar paint shop optimization with quantum annealing. In 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), pages 35–41. IEEE. https://doi.org/10.1109/qce52317.2021.00019

Zhang S-X, Wan Z-Q, Lee C-K, Hsieh C-Y, Zhang S, Yao H (2021) Variational quantum-neural hybrid eigensolver. Phys. Rev. Lett. 128:120502. https://doi.org/10.1103/PhysRevLett.128.120502

Zhao C, Gao X-S (2019) QDNN: DNN with quantum neural network layers. arXiv:1912.12660

Zheltkov D, Osinsky A (2020) Global optimization algorithms using tensor trains. Lecture Notes Comput Sci 11958:197–202. https://doi.org/10.1007/978-3-030-41032-2_22

Zheltkov D, Tyrtyshnikov E (2020) Global optimization based on TT-decomposition. Russian J Numerical Anal Math Modell 35(4):247–261. https://doi.org/10.1515/rnam-2020-0021