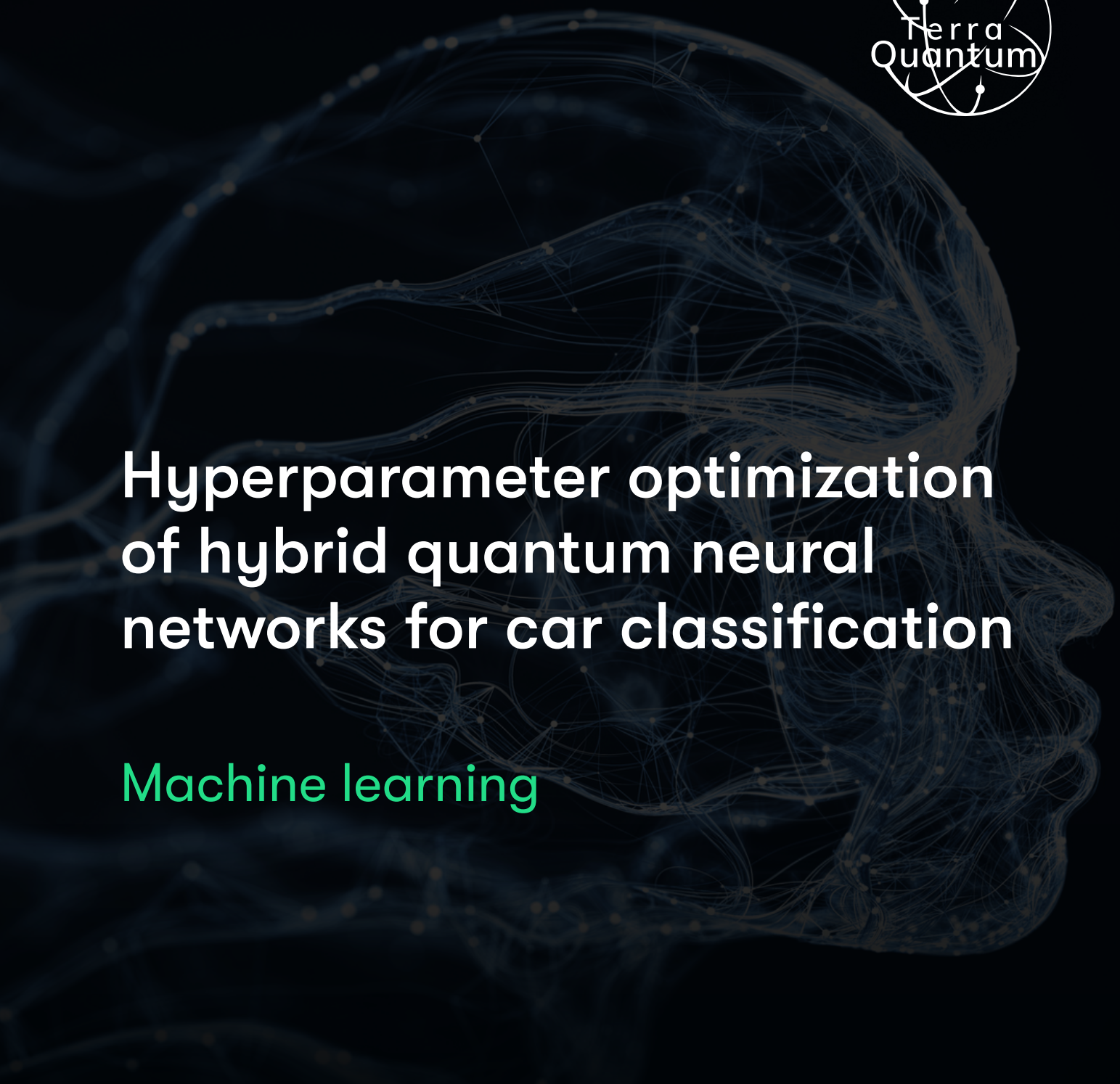

Hyperparameter optimization of hybrid quantum neural networks for car classification

Machine learning

By Terra Quantum AG
terraquantum.swiss

2022

Hyperparameter optimization of hybrid quantum neural networks for car classification

Asel Sagingalieva, Andrii Kurkin, Artem Melnikov, Daniil Kuhmistrov, Michael Perelshtein, and Alexey Melnikov
Terra Quantum AG, 9400 Rorschach, Switzerland

Andrea Skolik[1,3] and David Von Dollen[2,3]
[1] *Volkswagen Data:Lab, 80805 Munich, Germany*
[2] *Volkswagen Group of America, Auburn Hills, MI 48326, USA and*
[3] *Leiden University, 2333 CA Leiden, The Netherlands*

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 neural network 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.

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 [1]. 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 [2], which prompted a number of works by other automotive companies [3–5]. Since then, quantum annealing has been applied in a number of industry-related problems like chemistry [6, 7], aviation [8], logistics [9] and finance [10]. 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 [11]. 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 [12–16], quantum chemistry and simulation [17, 18], and machine learning [19–23]. Research in the industrial context has been largely motivated by noisy intermediate-scale quantum (NISQ) devices – 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 [24]. 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 [25–30]. 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-classical model to perform image classification on a subset of the Stanford Cars data set [31]. 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 [32–35], 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 [36]. While the separation in [36] 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 data sets.

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 data sets using best-first search and cross validation [37]. In more recent years, researchers have proposed search algorithms using bandits [38], which leverage early stopping methods. Successive Halving algorithms such as the one introduced in [39] and the parallelized version introduced in [40] 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 [41, 42]. 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 [43–45]. 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 [46] and automated machine learning (AutoML) [47]. Automating the search of hyperparameters in a quantum machine learning (QML) context has also started to attract attention, and the authors of [48] have explored the first version of AutoQML.

Our contribution in this paper is not only to examine the performance gains of hybrid quantum-classical 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 HQNN framed as a global optimization problem as in [49]. This method has been successful in optimizing models of social networks in [50], and as a method of compression for deep neural networks [51].

RESULTS

A. Hyperparameter Optimization

The problem of hyperparameter optimization (HPO) is described schematically in Fig. 1(a). Given a certain data set 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 $A(\bar{h}_{\text{opt}})$ with the corresponding hyperparameters $\bar{h}_{\text{opt}}$.

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.

B. 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 (DMRG) [52]. In this approach, the ground state is represented in the TT format, often referred to as the Matrix Product State in physics [53]. 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

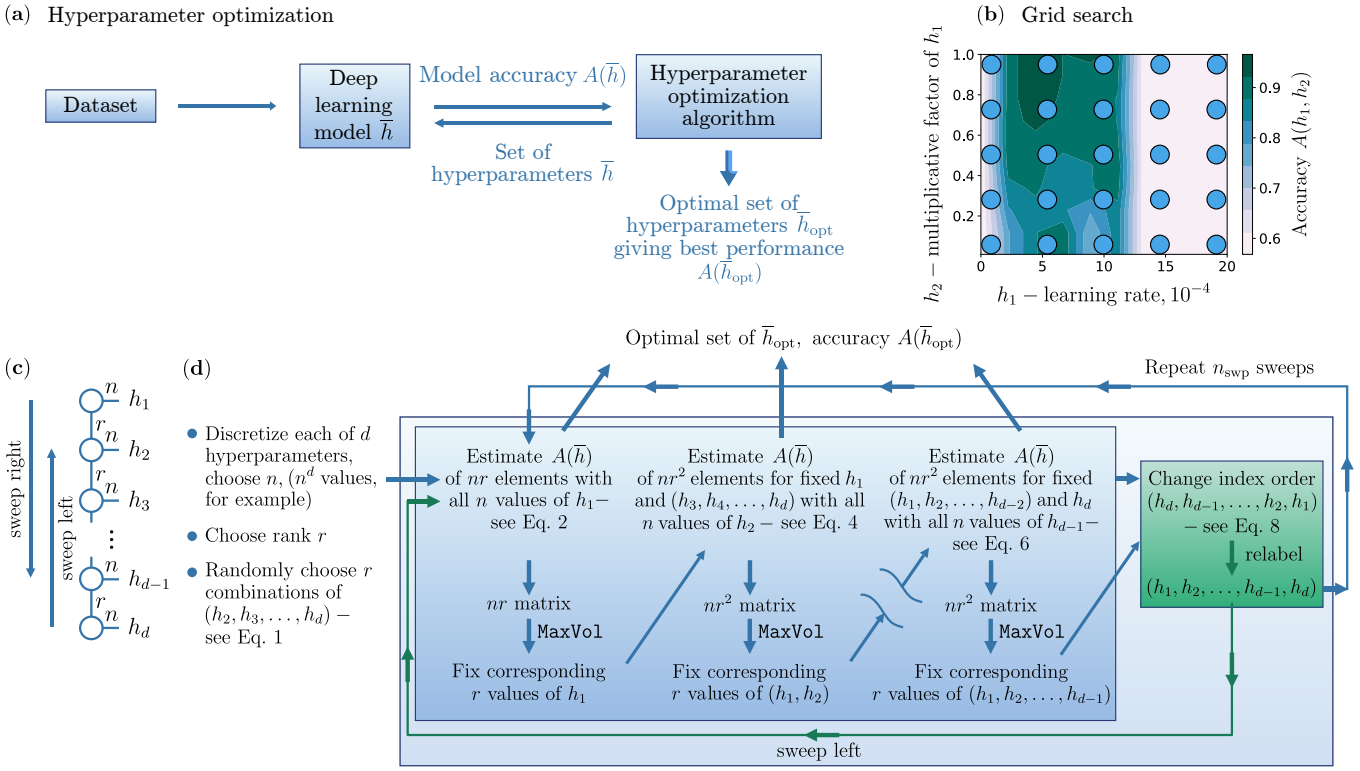

FIG. 1: The hyperparameter optimization problem description (a). The tabular methods for hyperparameter optimization: the grid search algorithm (b) and the tensor train algorithm (c-d).

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 [54, 55].

Similar to the previously discussed GS method, we discretize the hyperparameter space with TT optimization (TTO) 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 TTO 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 [54]. During the process, new sets of hyperparameters for which the model needs to be evaluated are determined using the `MaxVol` routine [56]. 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 TTO 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 TTO algorithm for the HPO is presented as the Algorithm 1 pseudocode that also corresponds to Fig. 1(d). The TTO 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
$$\{h_2^1(j), h_3^1(j), \ldots, h_d^1(j)\}_{j=1}^{j=r}. \quad (1)$$
are chosen.

Algorithm 1 Tensor Train Optimization

Accuracy $A(\bar{h}_{\text{opt}}) = 0$
$i_{\text{swp}} = 1$
Core $= 1$
$j_{\text{swp}} = 1$
Discretize each of d hyperparameters with n points
Randomly choose r combinations of $(h_2, h_3, \ldots, h_d)$
while $i_{\text{swp}} \leq n_{\text{swp}}$ **do**
  while $j_{\text{swp}} \leq 2$ **do**
    while Core $< d$ **do**
      if Core $==1$ **then**
        Estimate $A(\bar{h})$ of nr elements with all n values of h_1
        if $A(\bar{h}_{\text{opt}}) < A(\bar{h})$ **then**
          $A(\bar{h}_{\text{opt}}) = A(\bar{h})$
        end if
        `MaxVol`
        Fix corresponding r values of h_1
      else
        Estimate $A(\bar{h})$ of nr^2 elements for fixed $(h_1, \ldots, h_{\text{Core}-1})$ with all n values of h_{Core}
        if $A(\bar{h}_{\text{opt}}) < A(\bar{h})$ **then**
          $A(\bar{h}_{\text{opt}}) = A(\bar{h})$
        end if
        `MaxVol`
        Fix corresponding r values of $(h_1, \ldots, h_{\text{Core}})$
      end if
      Core $=$ Core $+ 1$
    end while
    Change index order $(h_d, \ldots, h_1)$
    Relabel $(h_1, \ldots, h_d)$
    Core $= 1$
    $j_{\text{swp}} = j_{\text{swp}} + 1$
  end while
  $j_{\text{swp}} = 1$
  $i_{\text{swp}} = i_{\text{swp}} + 1$
end while

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:

   - 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 hyperparameter $\{h_2(i_2)\}_{i_2=1}^{i_2=n}$:

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

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

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

   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 $\{(h_1^{d-2}(k), h_2^{d-2}(k), \ldots, h_{d-2}^{d-2}(k))\}_{k=1}^{k=r}$ 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 $\{h_{d-1}(i_d)\}_{i_d=1}^{i_d=n}$:

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

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

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

   r combinations of $\{(h_1^{d-1}(k), h_2^{d-1}(k), \ldots, h_{d-1}^{d-1}(k)\}_{k=1}^{k=r}$ 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 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:

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

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

   And then, continues from the fourth step of the TTO 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.

9. During all the iterations, we record it if we estimate a new maximum score.

C. Benchmarking HPO Methods

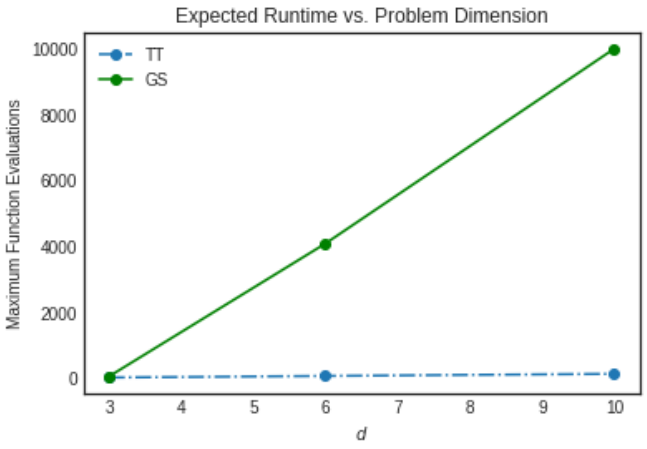

FIG. 2: Tensor Train (TT) and Grid Search (GS): Expected Runtime in maximum objective function evaluations vs. growth of problem dimension d.

Schwefel			
HPO Method	**Average Fitness**	d	*ER*
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	10000
Fletcher-Powell			
HPO Method	**Average Fitness**	d	*ER*
TT	5136.64	3	32
GS	**4113.78**	**3**	**64**
TT	23954.5	6	80
GS	**14295.2**	**6**	**4092**
TT	78101.4	10	144
GS	**36890.11**	**10**	**10000**
Vincent			
HPO Method	**Average Fitness**	d	*ER*
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**	**10000**

TABLE I: Table of results comparing HPO methods for Schwefel, Fletcher-Powell, and Vincent objective functions. 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*.

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 [57]. 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 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 $r = 2$.

D. 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 data set [58] 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 8,144 training images and 8,041 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 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 [59]. In our case, the ResNet (Residual Neural Network) [60] is pretrained on the ImageNet data set [61], 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 [62], 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 af-

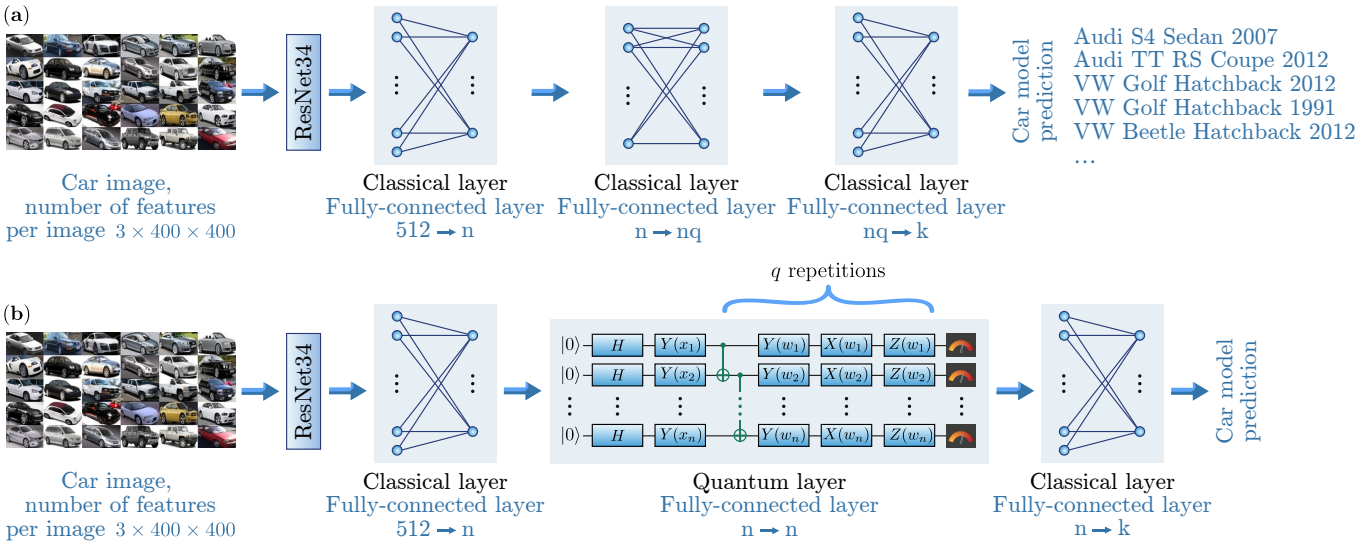

FIG. 3: Classical (a) and Hybrid (b) quantum neural network architectures.

ter 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 worth noting that the number of qubits used for the efficient operation of the model is initially unknown. 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, 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.

We use the cross-entropy as a loss function

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

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 run our model for 10 epochs and apply weight decay and gradient clipping to prevent interference from large gradient or weight values. We use the Adam optimizer [63, 64] and reduce the learning rate after several epochs. 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 II to improve the accuracy of solving the problem of car classification.

E. Simulation Results

We next perform a simulation of the hybrid quantum residual neural network 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,

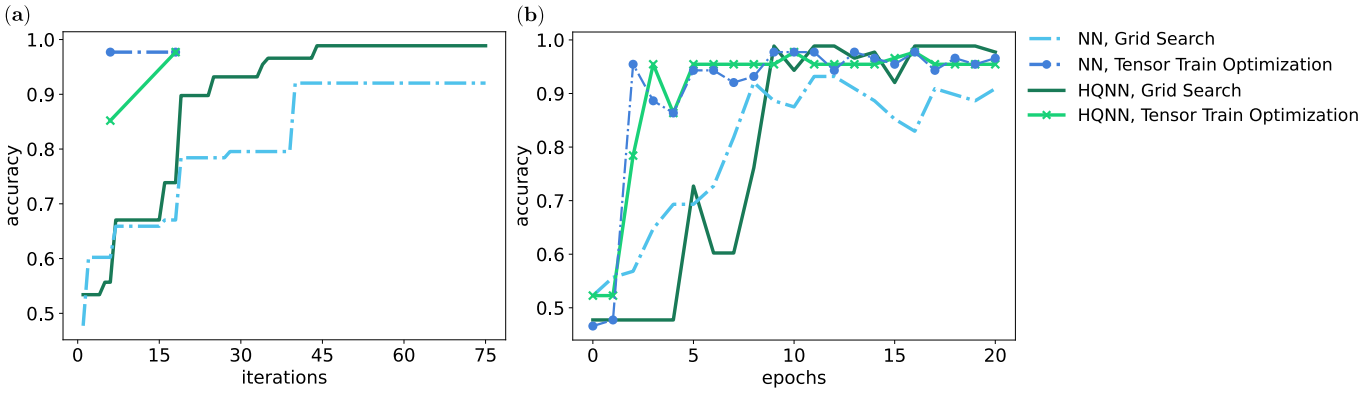

FIG. 4: (a) Dependence of accuracy on the number of iterations HPO. TTO for the hybrid model found a set of hyperparameters that gives an accuracy of 0.852 after 6 iterations, 0.977 after 18 iterations, for the classical model found 0.977 after 6 iterations. Grid search for the hybrid model found a set 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.

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 learning rate decay	α_r	$0.1-0.2$	0.1	0.2

TABLE II: The table shows which hyperparameters are being optimized, their labels, limits of change, and the best values found during HPO.

we used 88 testing images and 89 training images. Both the hybrid quantum HQNN model, and the classical NN model, were used together with the GS and TTO 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 [65]. The results of the simulations are shown in Fig. 4.

Fig. 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, blue color shows for the classical NN. As one can see from Fig. 4(a), TTO works more efficiently than GS and in fewer iterations finds hyperparameters that give an accuracy above 0.9. HQNN with TTO (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.

TTO 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.

FIG. 5: Examples of test car images that were correctly classified by the hybrid quantum residual neural network.

The best values found during HPO are displayed in Table II. 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 com-

pared 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%). The Fig. 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.

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 neural network 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 classification 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 TTO showed an accuracy of 0.977 after 18 iterations, whereas the comparable classical ML model with TTO, which 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 Ref. [66]. 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.

[1] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[2] Florian Neukart, Gabriele Compostella, Christian Seidel, David Von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in Information and Communication Technologies*, 4:29, 2017.

[3] Arpit Mehta, Murad Muradi, and Selam Woldetsadick. Quantum annealing based optimization of robotic movement in manufacturing. In *International Workshop on Quantum Technology and Optimization Problems*, pages 136–144. Springer, 2019.

[4] Masayuki Ohzeki, Akira Miki, Masamichi J Miyama, and Masayoshi Terabe. Control of automated guided vehicles without collision by quantum annealer and digital devices. *Frontiers in Computer Science*, 1:9, 2019.

[5] Sheir Yarkoni, Alex Alekseyenko, Michael Streif, David Von Dollen, Florian Neukart, and Thomas Bäck. Multi-car paint shop optimization with quantum annealing. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 35–41. IEEE, 2021.

[6] Michael Streif, Florian Neukart, and Martin Leib. Solving quantum chemistry problems with a d-wave quantum annealer. In *International Workshop on Quantum Technology and Optimization Problems*, pages 111–122. Springer, 2019.

[7] Rongxin Xia, Teng Bian, and Sabre Kais. Electronic structure calculations and the ising hamiltonian. *The Journal of Physical Chemistry B*, 122(13):3384–3395, 2017.

[8] Tobias Stollenwerk, Bryan O'Gorman, Davide Venturelli, Salvatore Mandra, Olga Rodionova, Hokkwan Ng, Banavar Sridhar, Eleanor Gilbert Rieffel, and Rupak Biswas. Quantum annealing applied to deconflicting optimal trajectories for air traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):285–297, 2019.

[9] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer. *Frontiers in Information and Communication Technologies*, 6:13, 2019.

[10] Erica Grant, Travis S Humble, and Benjamin Stump. Benchmarking quantum annealing controls with portfolio optimization. *Physical Review Applied*, 15(1):014012, 2021.

[11] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[12] Michael Streif, Sheir Yarkoni, Andrea Skolik, Florian Neukart, and Martin Leib. Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. *Physical Review A*, 104(1):012403, 2021.

[13] Michael Streif and Martin Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology*, 5(3):034008, 2020.

[14] David Amaro, Matthias Rosenkranz, Nathan Fitzpatrick, Koji Hirano, and Mattia Fiorentini. A case study of variational quantum algorithms for a job shop scheduling problem. *arXiv preprint arXiv:2109.03745*, 2021.

[15] Constantin Dalyac, Loïc Henriet, Emmanuel Jeandel, Wolfgang Lechner, Simon Perdrix, Marc Porcheron, and Margarita Veshchezerova. Qualifying quantum approaches for hard industrial optimization problems. a case study in the field of smart-charging of electric vehicles. *EPJ Quantum Technology*, 8(1):12, 2021.

[16] Andre Luckow, Johannes Klepsch, and Josef Pichlmeier. Quantum computing: Towards industry reference problems. *arXiv preprint arXiv:2103.07433*, 2021.

[17] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B Buckley, David A Buell, et al. Hartree-fock on a superconducting qubit quantum computer. *Science*, 369(6507):1084–1089, 2020.

[18] Fionn D Malone, Robert M Parrish, Alicia R Welden, Thomas Fox, Matthias Degroote, Elica Kyoseva, Nikolaj Moll, Raffaele Santagati, and Michael Streif. Towards the simulation of large scale protein-ligand interactions on nisq-era quantum computers. *arXiv preprint arXiv:2110.01589*, 2021.

[19] Manuel S Rudolph, Ntwali Bashige Toussaint, Amara Katabarwa, Sonika Johri, Borja Peropadre, and Alejandro Perdomo-Ortiz. Generation of high-resolution handwritten digits with an ion-trap quantum computer. *arXiv preprint arXiv:2012.03924*, 2020.

[20] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3(1):1–11, 2021.

[21] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *arXiv preprint arXiv:2103.15084*, 2021.

[22] Evan Peters, Joao Caldeira, Alan Ho, Stefan Leichenauer, Masoud Mohseni, Hartmut Neven, Panagiotis Spentzouris, Doug Strain, and Gabriel N Perdue. Machine learning of high dimensional data on a noisy quantum processor. *arXiv preprint arXiv:2101.09581*, 2021.

[23] Javier Alcazar, Vicente Leyton-Ortega, and Alejandro Perdomo-Ortiz. Classical versus quantum models in machine learning: insights from a finance application. *Machine Learning: Science and Technology*, 1(3):035003, 2020.

[24] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, pages 1–20, 2021.

[25] Shi-Xin Zhang, Zhou-Quan Wan, Chee-Kong Lee, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Variational quantum-neural hybrid eigensolver. *arXiv preprint arXiv:2106.05105*, 2021.

[26] Andrea Mari, Thomas R. Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran. Transfer learning in hybrid classical-quantum neural networks. *Quantum*, 4:340, 2020.

[27] Chen Zhao and Xiao-Shan Gao. Qdnn: Dnn with quantum neural network layers. *arXiv preprint arXiv:1912.12660*, 2019.

[28] Tong Dou, Kaiwei Wang, Zhenwei Zhou, Shilu Yan, and Wei Cui. 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, 2021.

[29] Alessandro Sebastianelli, Daniela Alessandra Zaidenberg, Dario Spiller, Bertrand Le Saux, and Silvia Liberata Ullo. On circuit-based hybrid quantum neural networks for remote sensing imagery classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:565–580, 2021.

[30] Sayantan Pramanik, M Girish Chandra, CV Sridhar, Aniket Kulkarni, Prabin Sahoo, Vishwa Chethan DV, Hrishikesh Sharma, Ashutosh Paliwal, Vidyut Navelkar, Sudhakara Poojary, et al. A quantum-classical hybrid method for image classification and segmentation. *arXiv preprint arXiv:2109.14431*, 2021.

[31] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[32] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[33] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122(4):040504, 2019.

[34] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[35] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014.

[36] 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.

[37] Ron Kohavi and George H. John. Automatic parameter selection by minimizing estimated error. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 304–312. Morgan Kaufmann, San Francisco (CA), 1995.

[38] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[39] Zohar Karnin, Tomer Koren, and Oren Somekh. 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, 2013. PMLR.

[40] Liam Li, Kevin G. Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*, 2018.

[41] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523. Springer Berlin Heidelberg, 2011.

[42] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[43] Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. Sequential vs. integrated algorithm selection and configuration: A case study for the modular cma-es. *arXiv preprint arXiv:1912.05899*, 2020.

[44] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Inc., USA, 1996.

[45] Noor Awad, Gresa Shala, Difan Deng, Neeratyoy Mallik, Matthias Feurer, Katharina Eggensperger, Andre' Biedenkapp, Diederick Vermetten, Hao Wang, Carola Doerr, Marius Lindauer, and Frank Hutter. Squirrel: A switching hyperparameter optimizer. *arXiv preprint arXiv:2012.08180*, 2020.

[46] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[47] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature, 2019.

[48] Raúl Berganza Gómez, Corey O'Meara, Giorgio Cortiana, Christian B. Mendl, and Juan Bernabé-Moreno. Towards autoqml: A cloud-based automated circuit architecture search framework. *arXiv preprint arXiv:2202.08024*, 2022.

[49] Dmitry Zheltkov and Alexander Osinsky. Global optimization algorithms using tensor trains. *Lecture Notes in Computer Science*, 11958:197–202, 2020.

[50] Sergey Kabanikhin, Olga Krivorotko, Shuhua Zhang, Victoriya Kashtanova, and Yufang Wang. Tensor train optimization for mathematical model of social networks. *arXiv preprint arXiv:1906.05246*, 2019.

[51] Dingheng Wang, Guangshe Zhao, Hengnu Chen, Zhexian Liu, Lei Deng, and Guoqi Li. Nonlinear tensor train format for deep neural network compression. *Neural Networks*, 144:320–333, 2021.

[52] Steven R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69:2863–2866, 1992.

[53] J. Ignacio Cirac, David Pérez-García, Norbert Schuch, and Frank Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, 93(4), 2021.

[54] Ivan Oseledets and Eugene Tyrtyshnikov. Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[55] Dmitry Zheltkov and Eugene Tyrtyshnikov. Global optimization based on tt-decomposition. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 35(4):247–261, 2020.

[56] Sergei Goreinov, Ivan Oseledets, D. Savostyanov, E. Tyrtyshnikov, and Nickolai Zamarashkin. How to find a good submatrix. *Matrix Methods: Theory, Algorithms and Applications*, 2010.

[57] `optproblems`. https://pypi.org/project/optproblems/, 2022.

[58] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[59] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[61] Imagenet dataset. https://image-net.org/, 2022.

[62] PyTorch. https://pytorch.org/, 2022.

[63] Adam optimizer. https://pytorch.org/docs/stable/generated/torch.optim.Adam.html, 2022.

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] QMware, The first global quantum cloud. https://qm-ware.com, 2022.

[66] 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 preprint arXiv:2111.05292*, 2021.