QUANTUM CIRCUIT DESIGN SEARCH

Mohammad Pirhooshyaran
ISE, Lehigh University
mop216@lehigh.edu

Tamás Terlaky
ISE, Lehigh University
terlaky@lehigh.edu

ABSTRACT

This article explores search strategies for the design of parameterized quantum circuits. We propose several optimization approaches including random search plus survival of the fittest, reinforcement learning and Bayesian optimization as decision makers to design a quantum circuit in an automated way for a specific task such as multi-labeled classification over a dataset. We introduce nontrivial circuit architectures that are arduous to be hand-designed and efficient in terms of trainability. In addition, we introduce reuploading of initial data into quantum circuits as an option to find more general designs. We numerically show that some of the suggested architectures for the Iris dataset accomplish better results compared to the established parameterized quantum circuit designs in the literature. In addition, we investigate the trainability of these structures on the unseen dataset Glass. We report meaningful advantages over the benchmarks for the classification of the Glass dataset which supports the fact that the suggested designs are inherently more trainable.

Keywords: Parameterized quantum circuits; Quantum circuit design; Circuit design search; Multi-labeled classification; Reinforcement learning; Random search

1 Introduction

In the era of noisy intermediate scale quantum (NISQ) [1] devices, variational quantum algorithms (VQA) [2] are promising tools to integrate classical machine learning methods with quantum frameworks. Parameterized quantum circuits (PQC) [3] are introduced to tune parameters of a quantum circuit by leveraging classical optimization methods. Variational eigenvalue solvers [4, 5] and quantum neural networks (QNN) [6–11] are among the many applications of PQC.

Analytical approaches towards calculating partial derivatives of the PQC outcome with respect to its parameters such as parameter-shift rule [3, 12, 13] pave the way for using gradient-based optimization methods to tune the circuit parameters.

However, recent studies identify barren plateaus and vanishing gradient problems related to the PQC training landscapes and cost-functions in QNNs [14, 15]. For a wide class of PQCs, random designs and/or defining cost functions in terms of global observables result in an exponentially vanishing partial gradients of the cost function with respect to the circuit parameters. Researchers suggest initialization [16], data reuploading [17] strategies and introduce cost functions in terms of local observables [15] to alleviate the negative effect of vanishing gradient but finding more efficient circuit designs has not been fully explored yet. In many cases, some established designs are used for PQC framework consisting of layers with a fixed pair of parameterized rotation operation such as R_y accompanied by a non-parameterized gate such as a controlled X or Z gates.

Recently Alexeev et al. [18] present a reinforcement learning (RL) method responsible for optimizing the quantum approximate optimization algorithm (QAOA) circuit parameters. In this study, we investigate the idea of the quantum circuit design search (QCDS) for a general supervised learning task with the hope of finding more efficient circuit architectures. We use three separate design search strategies. We implement a simple yet effective random search over the vast space of feasible designs combined with the survival of the fittest. Second, a policy gradient reinforcement learning is introduced due to the successful predecessor work in classical machine learning (ML) area [19, 20]. Third, a Bayesian black-box optimization (BO) is presented.

In this study, we aim to keep the number of parameters of PQCs fixed while trying to improve their test accuracy (loss) for the task of supervised ML. We discover and present many nontrivial circuit designs that are efficient enough to compete with a typical (rotation-controlled) designs in the literature. In addition, we tackle the task of multi-label classification by considering small to moderate datasets such as Iris and Glass. Glass dataset in particular consists of 6 separate labels. We do not limit the QCDS only to device-efficient operations that have already been applied successfully on real quantum devices. In other words, we are not necessarily concerned about the extra number of controlled gates or non-linear connections between qubits in a design but instead we suggest designs that are efficient in terms of their ability to be trained. We hope this shed some light on designing new circuits as well. The rest of the paper is as following. We discuss the problem and its background in the Section 2. Then, we present the search strategies in Section 3. The results will be reported in Section 4 and at last we present some extra discussion and conclusion in Section 5. In addition, parts of the implementation is available at the paper's github repository.

2 Theory and Background

In general, PQCs are favorable approaches because [6, 15]: First, one can theoretically design a task-specific PQC; Second, a PQC can benefit from new advances in classical paradigms at least in two ways, either by outsourcing the parameter update routine to a classical optimization algorithm or by using a PQC as only a component of a larger design, involving other classical parts; Third, these methods are more resistant against quantum noise, because the circuit parameters can learn the effect of the noise to some degree during the training; Forth, the PQC volume may be independent of or logarithmic in the size of the dataset they are dealing with [17]. That is, in contrast with many early quantum algorithms, such as Grover's algorithm which needs a depth of around $\sqrt{N}$, where N is the size of the dataset, in PQC, the depth of the parameterized circuit (i.e., the number of layers of the quantum operators stacked upon each-other) and the width of the circuit (the summation of all the input and ancilla qubits) can be a constant or relatively small number compared to the size of the dataset.

In this study, we consider the classification task of

$$\min_{\theta} J(\theta) := E\left(L\left(f\left(x;\theta\right), y\right)\right) \tag{1}$$

where the PQC function $f(x;\theta)$ is given as

$$f(x;\theta) = \langle 0|U(x;\theta)^{\dagger} H U(x;\theta)|0\rangle . \tag{2}$$

The quantum circuit $U(x;\theta)$ is parameterized over θ and receives x as an input feature. L is an appropriate loss function which in a sense compares the output of the PQC function f and the true labels y. The outer expectation is due to the stochasticity of the setting. This might come from the fact that we empirically evaluate the $f(x;\theta)$ at random subsamples x and never have access to the unknown distribution X where the input x is drown from or arise from some inherent noise. In addition, we have an implicit expectation over observable H at the end of the PQC, $\langle \psi, H\psi\rangle$ where $|\psi\rangle = |U(x;\theta)|0\rangle$. Unitary $U(x;\theta)$ can be written as a chain of unitary transformations [3, 15, 16]. We propose

$$U(x;\theta) = \prod_{l=0}^{L-1} \bigotimes_{i=0}^{N-1} \left(T_i^l\left(x_i^0\right) U_i^l\left(\theta_i^l\right) W_i^l\right) \tag{3}$$

as a general design of the PQC where N is the number of qubits and L is the number of the layers. Superscripts represent layers while subscripts represent qubits. x_i^0 is a subset of input features x where at the beginning has been injected for the first time into the qubit i of the circuit. $T_i^l\left(x_i^0\right)$ is an option of reuploading x_i^0 for the qubit i at layer l. We assume, either $T_i^l\left(x_i^0\right) = R_y\left(x_i^0\right)$ meaning that for the layer l we reupload the initial features via a rotation gate around y-axis or $T_i^l\left(x_i^0\right) = I$ where I is an appropriate identity matrix meaning that we reject the feature reuploading for layer l.

We assume $U_i^l\left(\theta_i^l\right) = exp\left(-i\theta_i^l V_i^l\right)$ where V_i^l is a Hermitian operator, particularly we consider Pauli matrices σ_x, σ_y and σ_z as possible choices for V_i^l. W_i^l are the nonparametric gates and we consider Pauli matrices, Hadamard, CNOT, CSWAP and Toffoli choices. Please refer to Figure 10 to see two examples of the above expressions.

Similar designs to the ones discussed in the literature [15, 16] can be extracted as special cases of equation (3) by turning off the reuploading option, choosing $U_i^l\left(\theta_i^l\right) = R_y\left(\theta_i^0\right)$ and a CZ as W_i^l for all qubits $i \in \{0, 1, \ldots N-1\}$ and $l \in \{0, 1, \ldots L-1\}$.

We consider

$$H = \bigotimes_{i \in \left[\bar{N}\right]} \sigma_z \bigotimes_{i \in [N] \setminus \left[\bar{N}\right]} I \tag{4}$$

as the observable in equation (2) where $[N] = \{0, 1, \ldots, N-1\}$ and $\left[\bar{N}\right] = \{0, 1, \ldots, \bar{N}-1\}$ is the set of all qubits that will be measured. Note that N and $\bar{N}$ (the number of input qubits and the number of output measurements) are not necessarily the same.

As long as Hermitian operators generating unitaries of the parametric gates, have two unique eigenvalues, which is the case for Pauli matrices, parameter-shift rule applies for partial derivative calculations [12]. For the sake of presenting the partial derivatives of $f(x;\theta)$ with respect to its parameters, we modify equation (3) as

$$U(x;\theta) = \prod_{l=0}^{L-1} \prod_{i=0}^{N-1} \left(\tilde{T}_i^l \left(x_i^0\right) \tilde{U}_i^l \left(\theta_i^l\right) \tilde{W}_i^l \right) \tag{5}$$

where

$$\begin{aligned} \tilde{T}_i^l &= \bigotimes_{j \in [N] \setminus i} I \bigotimes T_i^l \\ \tilde{U}_i^l &= \bigotimes_{j \in [N] \setminus i} I \bigotimes U_i^l \\ \tilde{W}_i^l &= \bigotimes_{j \in [N] \setminus i} I \bigotimes W_i^l. \end{aligned} \tag{6}$$

Then, we have

$$\frac{\partial f(x;\theta)}{\partial \theta_j^k} = i \langle 0 | \tilde{U}_-^\dagger \left[V_j^k, \tilde{U}_+^\dagger H \tilde{U}_+ \right] \tilde{U}_- | 0 \rangle \tag{7}$$

where $\tilde{U}_- = \prod_{l=0}^{k} \prod_{i=0}^{j-1} \left(\tilde{T}_i^l \left(x_i^0\right) \tilde{U}_i^l \left(\theta_i^l\right) \tilde{W}_i^l \right)$ and $\tilde{U}_+ = \prod_{l=k}^{L-1} \prod_{i=j}^{N-1} \left(\tilde{T}_i^l \left(x_i^0\right) \tilde{U}_i^l \left(\theta_i^l\right) \tilde{W}_i^l \right)$.

The PQC architecture can be used as a part of a larger architecture including other classical and/or quantum parts. The input feature x can be the output of another training architecture. Particularly, many large machine learning datasets can be down-scaled to a feature size that today's quantum circuits can handle. Moreover, instead of going through the loss function, the output of PQC $f(x;\theta)$ can continue to be fed into another architecture as an input feature. As long as we consider PQC as the last architecture, we assume the number of qubits involved in the output measurements ($\bar{N}$) equals to the number of classes and then the output of the PQC goes through the negative log-likelihood loss function. In the following section we discuss the approaches we use to search for the PQC architectures.

3 Design Search

The QCDS can be modeled as an optimization problem

$$\arg\min_{d} \min_{\theta} J(\theta); \text{ s.t. } d \in D \tag{8}$$

where $J(\theta)$ was introduced in equation (1) and d is a design drawn from D, the set of all the feasible designs, to identify the exact operations taken by $U(x;\theta)$ in equation (3). QCDS consists of three parts: establishing the search space D, proposing search strategy, and performance metric. In this study, we assume single qubit search space. In other words, we look at the PQC designs from a micro-level point of view. Each qubit can accept or reject the reuploading of the data it observes at the beginning of the circuit. It can then choose any rotation in $\{R_x, R_y, R_z\}$ and at last it can choose a nonparametric gate in $\{\text{H}, \sigma_x, \sigma_y, \sigma_z, \text{CNOT}, \text{CSwap}, \text{Toffoli}, \text{CZ}\}$. For a simple layer of 4 qubits the search space passes more than 5 million designs and by stacking-up layers of qubits, we make $|D|$ exponentially larger. Another approach for building a QCDS search space is to consider blocks of operations governing on more than one qubit. But this approach requires strong and efficient quantum blocks which brings us back to the micro-level approach.

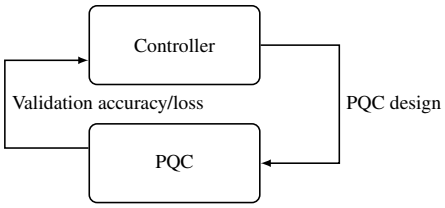

Figure 1: Controller-PQC interaction: the controller provide the PQC with a design and receives back the train accuracy (loss). Then, it updates its parameters by policy gradient RL method considering the PQC train report plus the suggested design entropy and policy distribution.

For the search strategy we assume three methods, reinforcement learning, random search and Bayesian optimization. We go through them in the rest of this section. For the performance metric we consider either the accuracy of the validation dataset or its loss value.

3.1 Reinforcement learning for QCDS

In this section we focus on reinforcement learning for quantum circuit design search (R-QCDS). We introduce an environment similar to the classical machine learning work [19, 20] in which a controller and a PQC interact with each other to find the best design for the PQC. We propose a deep neural network (DNN) as the controller. At each time-step t controller outputs a policy (joint distribution) $\Pi_t(w)$ over all the decisions to be made, parameterized by w the weights of the DNN. Then, a sample is drwan from $\Pi_t(w)$ as the suggested design. The PQC trains based on the design over training set and then reports back the accuracy (loss) over the validation set to the controller. The controller, then, optimizes its policy and updates its own parameters w using the PQC feedback as well as characteristics of the distribution $\Pi_t(w)$ such as entropy. To do the update, controller uses the policy gradient method [21].

Figure 1 illustrates the environment interaction. Either controller suggests the entire design of the QPC or it only suggests the design of a layer which we then repeat its pattern throughout the entire PQC. Figure 2 shows how controller presents the distribution over the decisions related to a single node (qubit). There are three decisions to make, uploading the data, parametric operation, nonparametric operation. In addition, PQC can send back the validation loss as the loss or validation accuracy as the reward to the controller. Therefore, we have two types of policies and two types of performance metric resulting in the total of 4 different R-QCDS approaches. When we consider accuracy as the performance metric, as the loss for the controller we assume (1-accuracy).

We use a feed-forward DNN in contrast to RNN networks used in [19, 20]. Therefore, there is no input to the controller. The controller has two shared layers (of the sizes of 48×12 in our study) and a specific layer for each of the reuploading, parametric and nonparametric gate decisions. The output layers provide us with the distributions over the corresponding decisions. Therefore, the number of nodes in the output layers matches the possible outcomes of corresponding decision. For instance, in the case of reuploading decision, the controller output is a Bernoulli distribution. Then, a sample is drawn and the desired action is suggested for the reuploading decision.

3.2 Random Search

Arguably the most simple yet powerful approach to search for a PQC design is a random search. We choose a large set of designs D randomly. We further make sure that the selected designs are different enough from one another. That is, for every $d_1, d_2 \in D$, we check that their sequence of decisions match below a threshold considering Gestalt pattern matching [22]. We train all the designs for a very few epochs (in our numerical cases two) and keep some portion of the designs based on their loss (in our numerical cases the better half). Then, we train the remaining for few more epochs (five epochs in our studies) and repeat the process until we reach one of the following criteria: 1) We reduce the size of the remaining designs to a number where we can train them all to their best ability; 2) The number of training epochs experimentally reaches the best possible value; 3) We realize the rankings of the remaining designs with respect to their losses are not changing dramatically with increasing the training epochs after some initial steps.

At each layer for a single qubit we consider $2 \times 3 \times 8 = 48$ possible outcomes having reuploading, parametric and nonparametric operations decisions. Imagine a PQC has 6 layers of 4 qubits. The random search space ends up to have more than 10^{30} designs. We consider validation loss values for comparison between the random designs to rank them. One can alternatively use classification accuracy as the comparison metric.

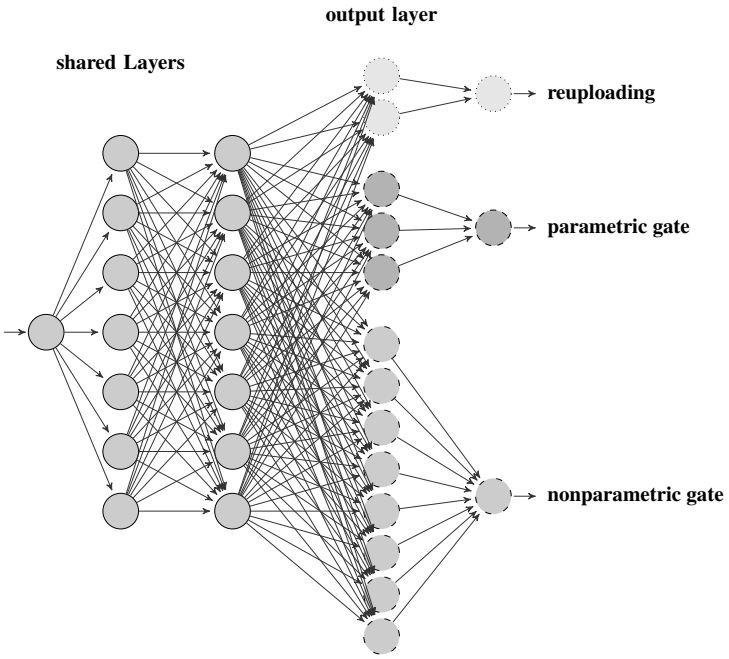

Figure 2: Controller network structure: a deep neural network consisting of two shared layers for all the decisions plus a layer corresponding to each decision is considered. The output of the controller is a joint distribution of all the decisions from which we sample to find the design.

3.3 Black-Box Bayesian Optimization

We focus on Bayesian optimization (BO) in order to suggest new PQC designs. In this approach, we consider the PQC as a black-box function which we only have access to its objective function evaluations and we assign a prior Gaussian distribution to every PQC decisions' distribution. Gaussian processes inherently consider the decision variables to accept real values but here all the decision variables are discrete. We consider a hyper-rectangle $\{x \in \mathbb{R}^d | l_i \leq x_i \leq u_i\}$ as the feasible set of all decision variables with l_i and u_i be the lower and upper bounds for decision variable x_i. We divide the interval $[l_i, u_i]$ into unit length sub-intervals and assign one unique integer outcome to any of these sub-intervals similar to literature [23, 24]. For instance, for x_i representing a parametric gate option we set $l_i = -\frac{1}{2}$ and $u_i = \frac{5}{2}$ and then divide the interval into three sub-intervals as $\left[-\frac{1}{2}, \frac{1}{2}\right]$, $\left[\frac{1}{2}, \frac{3}{2}\right]$, $\left[\frac{3}{2}, \frac{5}{2}\right]$ representing R_x, R_y and R_z respectively.

Figure 3 illustrates the general BO process we use. We consider validation loss values as the comparison metric for BO to evaluate. We use logarithmic expected improvement (Log EI) as the acquisition function [25]. That is, we aim to select next design so that the reduction over the validation loss becomes as large as possible; however, any actual reduction is unknown until after next examination. Therefore, we instead calculate the logarithmic expectation of the reduction and select the argmax of this expression.

4 Results

In this section, we provide the numerical results and comparisons. During the experiments, we divide the datasets into three parts of train, validation and test unless otherwise mentioned. Models never see the test part. There are classical hyperparameters to be tuned in the learning process as well such as learning rate, mini-batch size. R-QCDS, also consists of other parameters such as controller learning rate, mini-batch size and entropy coefficient. To tune these parameters, we mainly use a very simple grid search. Throughout the numerical studies we consider the number of the circuit qubits to be equal to the features of the dataset. All the results are simulated via Pennylane quantum machine learning library [26].

In the following numerical studies we mainly use 6 quantum layers defined in the form of equation (3) stack upon each other. We use two datasets of Iris and Glass from `LIBSVM` repository. Both consist of multiple classes. Glass data points contain 9 features and 6 labels while Iris data points contain 4 features and 3 labels. First, we aim to show the

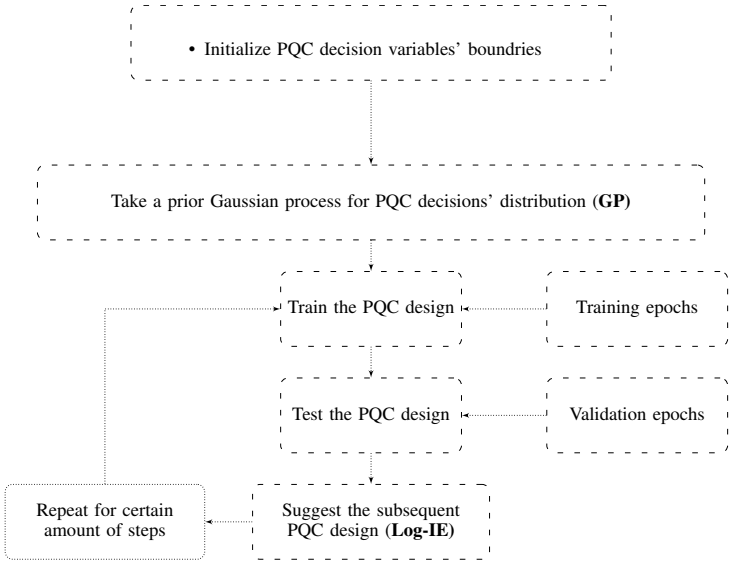

Figure 3: Gaussian process logarithmic expected improvement (GPLEI) Bayesian optimization method

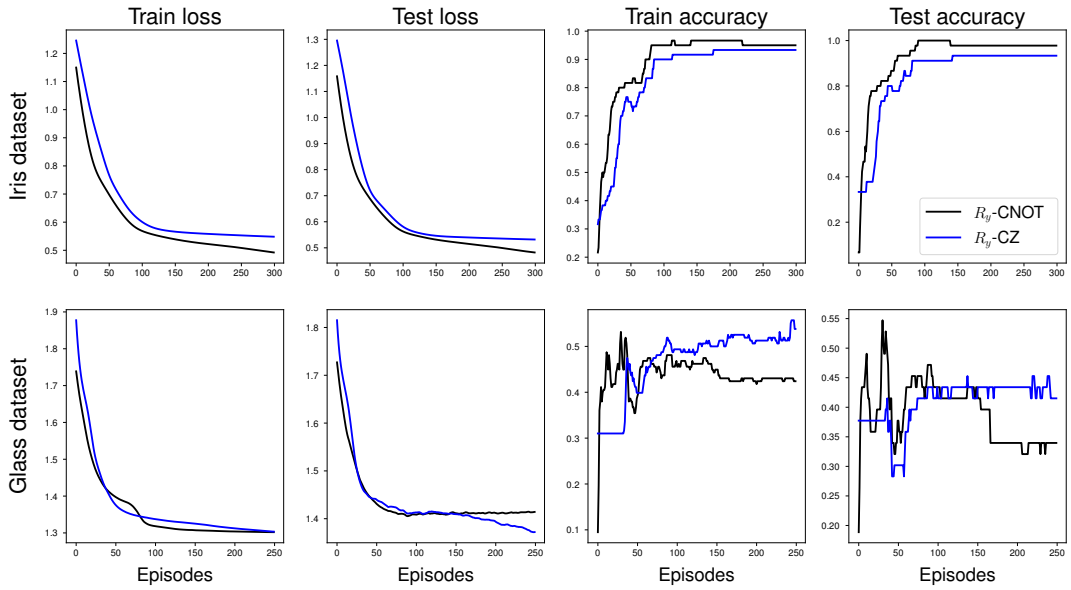

Figure 4: Train loss, train accuracy, test loss and test accuracy comparisons for (R_y rotation + CNOT) and (R_y rotation + CZ) PQC designs on Iris and Glass datasets. (R_y rotation + CNOT) design performs marginally better and hence would be considered as the benchmark for future comparisons. It reaches the test accuracy of 54.7% for the Glass dataset utilizing 6×9 parametric R_y gates.

validity of our modeling by illustrating the performances of two already established designs in the literature. Hence, we consider a pair of a single rotation R_y plus CNOT and a single rotation R_y plus CZ for each qubit [14–16]. Figure 4 shows the performances of the two designs. The train and test losses are divided by the size their datasets (and for the rest of the studies as well). For the Iris dataset the (R_y rotation + CNOT) design reaches 100% test accuracy at around 100 epochs into the training process. We can also see that (R_y rotation + CNOT) design works marginally better than (R_y rotation + CZ).

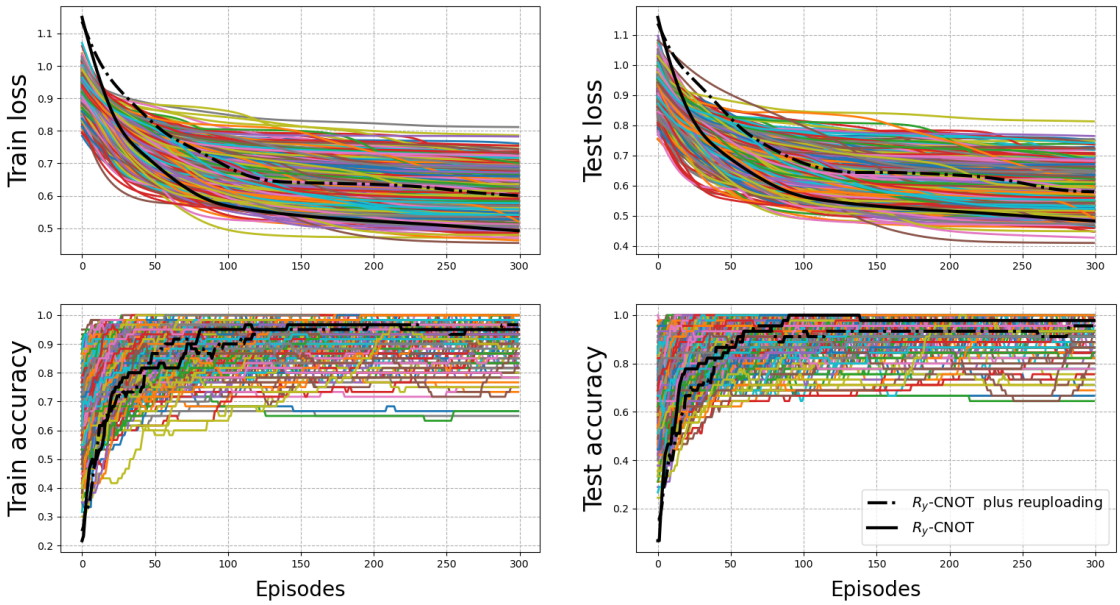

Figure 5: Train loss, train accuracy, test loss and test accuracy comparisons between the best 1000 designed found by random search and R_y rotation + CNOT design. One can certainly see some designs outperform the benchmark. In terms of test loss comparison, 2.5% of the set of the best 1000 random designs perform considerably better than benchmark.

We, therefore, consider (R_y rotation + CNOT) as the benchmark for comparison with designs extracted from the three proposed QCDS methods. First, we consider the random search method. We create 30 thousand randomly chosen designs. We do not allow designs to posses sequences matched more than 75%. We train them only for two epochs and rank them based on validation loss and throw away the worst half. Subsequently, we repeat the process considering the remaining designs trained for 5 epochs and then for 10 epochs. After reaching this point, we realize that the ranking of the best 1000 random designs doesn't change dramatically. Hence, we stop the process and train all the best 1000 designs for 300 epochs. Figure 5 illustrates the comparison between the benchmark design and the 1000 best designs found by the random search on the Iris dataset. We consider 40% training, 30% validation and 30% test.

As can be seen in Figure 5, there are many designs fail to produce competitive results but some of the discovered architectures perform considerably better than the (R_y rotation + CNOT) design in terms of train and test losses. Although the benchmark reaches the test accuracy of 100% as discussed before, many randomly discovered designs reach the same accuracy much quicker and also these designs are mainly robust enough to stay at 100% accuracy unlike the benchmark for which the test accuracy drops slightly. Considering that we conducted this study on a small- to moderate-sized dataset where the benchmark by itself reaches 100% accuracy as well as we only created 30k designs which is very tiny fraction of the designs for the considered PQC, this results reassure that there are many powerful and trainable designs in the search space.

We also investigate the performance of the benchmark when at each layer for all the qubits, we reupload the initial data. This has been shown as R_y-CNOT plus reuploading in Figure 5. One can see that this design performs inferior compared with the benchmark which in a sense emphasizes the fact that stacking up more gates without careful thought or any optimizing mechanism can easily result in less trainable quantum designs. One should only consider the introduced reuploading strategy as an option which based on an optimization strategy can be switched off or on for different qubits in different layers.

Now, we focus on the R-QCDS approaches discussed in the Section 3.1. For each feedback loop we train the suggested PQC from scratch for 50 epochs. After less than 50 feedback loop (depending on how large the controller learning rate is), controller's suggestions converge to a unique design. Figure 6 illustrates the comparison between 4 separate RL schemes and the (R_y rotation + CNOT). The Policy gradient methods are stable in terms of train and test accuracies but they are marginally inferior in terms of test loss in comparison with the benchmark.

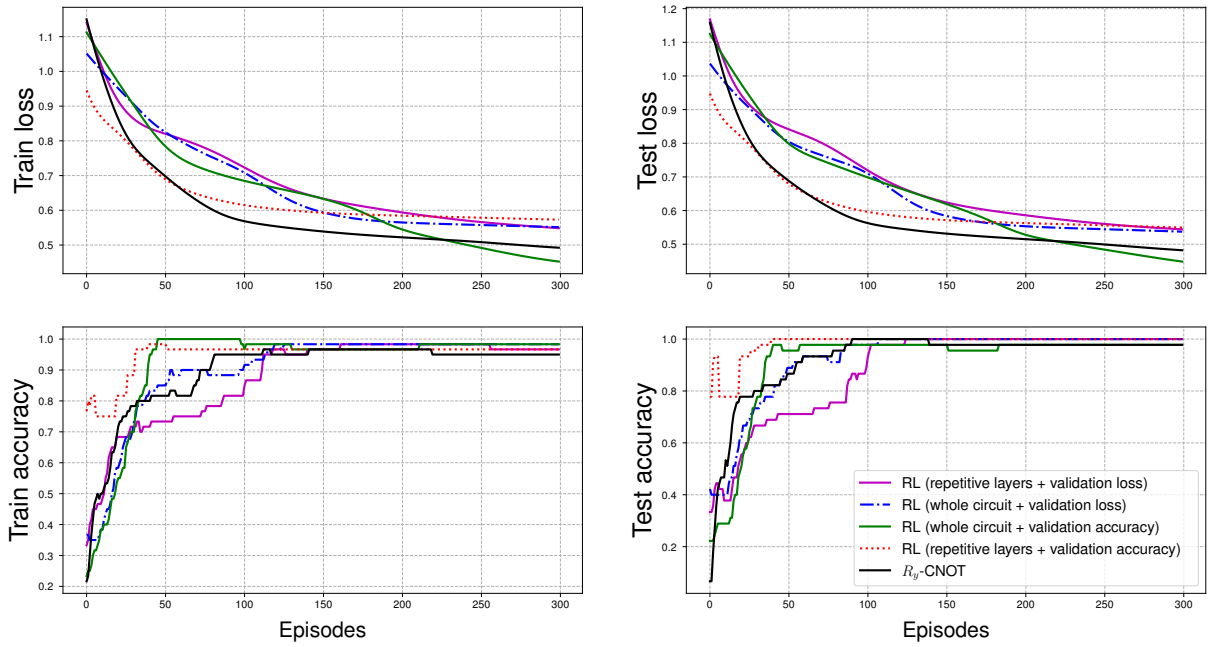

Figure 6: Train loss, train accuracy, test loss and test accuracy comparisons between R-QCDS strategies and typical R_y rotation + CNOT design. We consider four R-QCDS approaches. We use validation loss or validation accuracy as the performance metric. Moreover, we allow the R-QCDS methods to suggest only a quantum layer or the whole circuit. Controller learning rate is 0.1, 0.2, 0.02 and 0.02 for the different R-QCDS approaches in the order written in the plot legend.

Figure 7 illustrates the controller's learning curve on the validation dataset. For each point shown in the plots, a design is sent to the PQC, PQC is trained for 50 epochs and then the validation loss or accuracy on the validation dataset is fed back to the controller, then the policy plus entropy loss of the controller based on policy gradient RL method is calculated and plotted.

At last, we discuss the BO results. As we mentioned earlier, more accurate the segregate function results in better performance of BO. Hence, we train the PQCs for 100 epochs at each time step to then provide the validation loss. After around 70 function evaluations, BO suggests a design which performs superior to the (R_y rotation + CNOT) design. So, we stop the process. Figure 8 illustrates the comparison.

4.1 Generalization

The previous findings on Iris dataset confirm the importance of the QCDS. The superior designs in terms of trainability may inherently be more efficient than the benchmark and therefore extremely desirable for other applications and/or datasets. So, we go one step further and aim to compare the most efficient designs found in previous section with the benchmark on Glass dataset. We put an emphasis on the fact that the selected designs are totally agnostic towards Glass dataset. In addition, Glass dataset consists of 9 features for each data point and we assign a QPC of side 9 qubits to it. However, the architectures introduced previously consist of four qubits. So, we simply repeat the pattern to fully cover the Glass PQC. For instance, considering $0, \ldots, 8$ qubits used in the PQC for the Glass dataset, 8th and 4th qubits have similar operations as 0th qubit has with new tunable parameters.

We consider randomly chosen designs that performed considerably better than benchmark plus designs suggested by four different R-QCDS strategies plus BO. In this experiment, we split the data into 75% train and 25% test. No validation dataset is needed because the designs are already fixed. Methods never see the test dataset during the training. Although selected architectures were solely designed based on their performance on Iris dataset, the results shown in 9 strongly suggest they perform meaningfully better than the benchmark on another unseen dataset. The

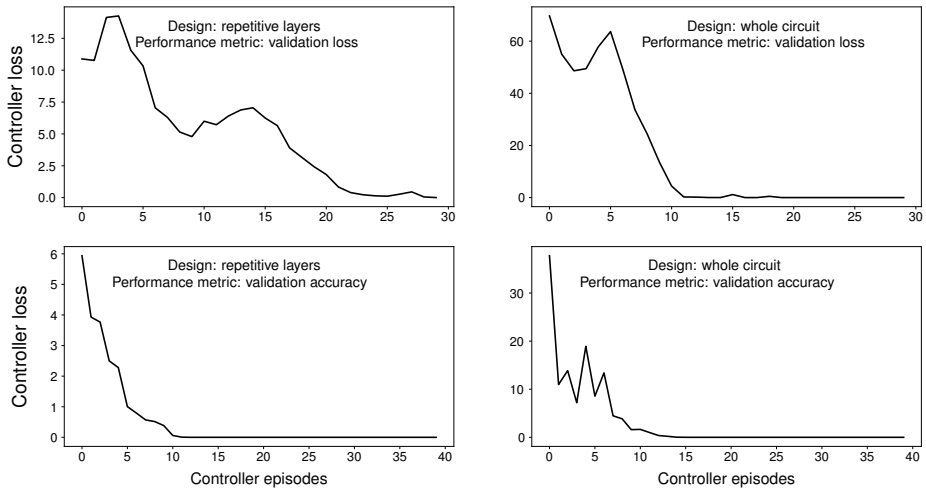

Figure 7: Controller's learning curve for 4 different R-QCDS strategies on the validation dataset.

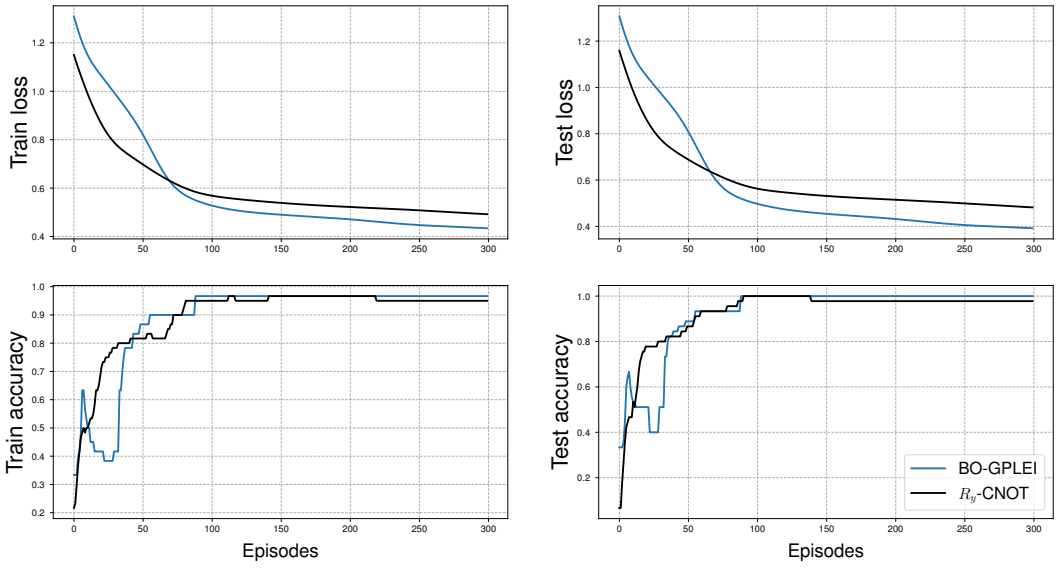

Figure 8: Train loss, train accuracy, test loss and test accuracy comparisons between GPLEI Bayesian optimization and R_y rotation + CNOT design.

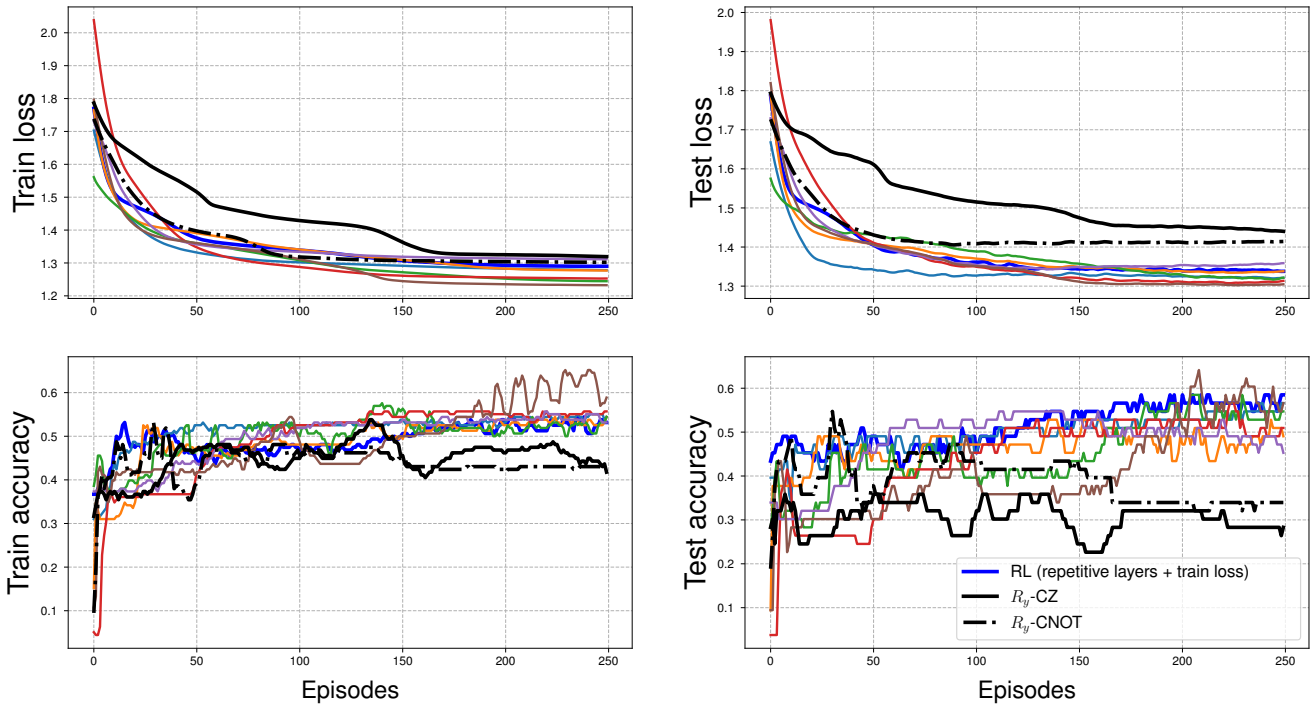

Figure 9: Train loss, train accuracy, test loss and test accuracy comparisons between selected designs and the benchmark design on Glass dataset. The second best test accuracy result is achieved by the design discovered by R-QCDS (design: repetitive layers plus performance metric: validation loss). The best accuracy reported is 64.1% from a randomly discovered design which presents 17.2% improvement over the benchmark design.

second best test accuracy result is achieved by a design discovered by R-QCDS (repetitive layers + train loss) and the best test accuracy result is achieved by one of the randomly discovered designs. The best test accuracy reported is 64.1% which means 17.2% improvement over the benchmark design. Figure 10 illustrates these two designs.

5 Discussion and Conclusion

This study is one of the first research which focuses on QCDS framework and investigate trainability of PQCs. Three optimization strategies are provided to find more efficient designs: random search plus survival of the fittest; reinforcement learning and Bayesian optimization. We suggest nontrivial automated designs which are hard to be hand-designed. We investigate their performances on unseen dataset and compare the results with benchmark PQC designs. One can see the strong impact of optimization strategies in designing of PQCs.

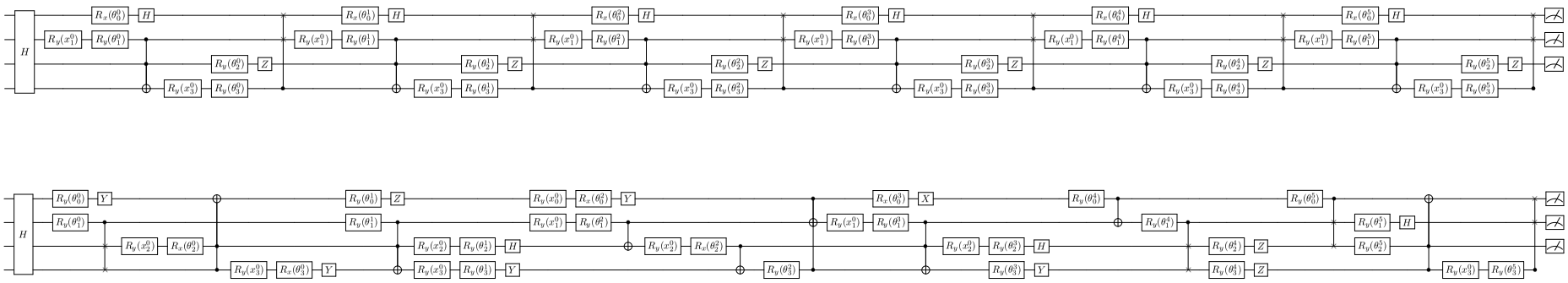

Figure 10: The best two designs reported in Figure 9. The top section design is suggested by R-QCDS (design: repetitive layers plus performance metric: validation loss) while the bottom one is one of the randomly chosen designs which performed better than the benchmark on the Iris dataset. In terms of test accuracy they outperform the benchmark by 6.9% and 17.2% margin, respectively. The designs are originally optimized to be used on Iris dataset. So, readers can see PQCs consists of 4 qubits with three single qubit σ_z measurements at the end.

References

[1] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[2] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[3] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

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

[5] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[6] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

[7] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1):1–8, 2018.

[8] Guillaume Verdon, Jason Pye, and Michael Broughton. A universal training algorithm for quantum deep learning. *arXiv preprint arXiv:1806.09729*, 2018.

[9] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *arXiv preprint arXiv:2006.14904*, 2020.

[10] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature communications*, 11(1):1–6, 2020.

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

[12] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[13] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[14] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[15] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost-function-dependent barren plateaus in shallow quantum neural networks. *arXiv preprint arXiv:2001.00550*, 2020.

[16] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[17] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[18] Yuri Alexeev, Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, and Prasanna Balaprakash. Reinforcement learning for finding qaoa parameters. *Bulletin of the American Physical Society*, 65, 2020.

[19] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[20] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[21] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[22] John W Ratcliff and David E Metzener. Pattern-matching-the gestalt approach. *Dr Dobbs Journal*, 13(7):46, 1988.

[23] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[24] Mohammad Pirhooshyaran, Katya Scheinberg, and Lawrence V Snyder. Feature engineering and forecasting via derivative-free optimization and ensemble of sequence-to-sequence networks with applications in renewable energy. *Energy*, 196:117136, 2020.

[25] A. Klein, S. Falkner, N. Mansur, and F. Hutter. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian Optimization Workshop*, December 2017.

[26] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.