Quantum Machine Intelligence (2025) 7:53
https://doi.org/10.1007/s42484-025-00277-1

RESEARCH ARTICLE

# A case study for cyber-attack detection using quantum variational circuits

**Maximilian Moll**[1] · **Leonhard Kunczik**[2]

Received: 28 December 2022 / Accepted: 1 April 2025


**Abstract**
Identification of malicious attacks in network traffic is one of many important big data problems that have been approached from different directions, including machine learning. In this paper, it is used as an example for investigating the applicability of quantum machine learning to such problems. Particular focus is kept on the NISQ era, in which available computer sizes are typically fairly small. Instead of applying typical cutting and knitting techniques with their associated overhead cost, we use the trainable output layer inherent in many hybrid QML approaches to re-combine the results from a collection of smaller QVCs executed on different machines. It is compared to classical and established quantum approaches on four real-world datasets.

**Keywords** Quantum machine learning · Quantum variational circuits · Cyber-attack detection · Gate-based quantum computers

## 1 Introduction

Within today's interconnected world, an abundance of network flow data exists and is continued to be created every second. A great challenge is to detect malicious packages within the network traffic to prevent cyberattacks (Oliveira et al. 2021). This big data problem (Wang and Jones 2021) has been studied with classical machine learning (ML). In this paper, however, we investigate whether and how such problems can be approached with quantum machine learning (QML).

The release of IBM's quantum hardware to the public in 2016 boosted the research in quantum computing (QC), and a journey to explore possible applications of QC in the future began. The main research areas of QC are cryptography, simulation (mainly of physical processes), optimization, and ML. While the first three concerns speeding up computations, the direct benefit of fusing QC and ML to QML is not fully understood (Schuld and Killoran 2022). However, research indicates that QML models can achieve the same results as their classical counterparts while utilizing smaller models, meaning less trainable parameters (Chen et al. 2020; Griol-Barres et al. 2021).

Within QML, multiple algorithms have been developed and successfully applied, including quantum support vector machines (Rebentrost et al. 2014; Havlíček et al. 2019) and quantum reinforcement learning (Dong et al. 2008; Chen et al. 2020; Meyer et al. 2022). This work focuses on quantum variational circuits (QVC), otherwise known as parameterized quantum circuits (PQC), which are either used to implement a quantum counterpart to classical neural networks (NN) or to combine them with NN to quantum-hybrid models. They have already shown great success in different applications (Azevedo et al. 2022; Mari et al. 2020). However, due to the limited size and quality of quantum computers available, the considered problems are usually small

Maximilian Moll and Leonhard Kunczik contributed equally to this work.

✉ Maximilian Moll
maximilian.moll@unibw.de

Leonhard Kunczik
leonhard.kunczik@unibw.de

[1] Institut für Theoretische Informatik, Mathematik und Operations Research, Universitat der Bundeswehr Munchen, Werner-Heisenberg-Weg 39, Neubiberg 85577, Bavaria, Germany

[2] Research Institute Cyber Defence (CODE), Universitat der Bundeswehr Munchen, Werner-Heisenberg-Weg 39, Neubiberg 85577, Bavaria, Germany

Published online: 12 April 2025

in nature. Thus, the question remains how they can be scaled to big data scenarios.

One approach to reduce the computational complexity, by which we mean the number of qubits and operations involved, is cutting the quantum circuit into smaller subcircuits (Dunjko et al. 2018; Peng et al. 2020). These circuits can be evaluated in parallel to reduce the problem to multiple smaller quantum computers instead of one larger system. Furthermore, performing multiple computations in parallel can be beneficial in terms of runtime. However, circuit cutting comes with a computational overhead, since it requires recomputing multiple circuits and inserting the results into other computations.

In this work, we introduce the idea of stacking multiple QVC in one model to overcome the limitations of current machines. Although our approach shares conceptual similarities with circuit cutting and knitting, it relies solely on ideas from QVC and classical machine learning. Furthermore, no additional classical computations are required, and it can easily be run in parallel on multiple quantum processors.

To show the potential benefits of this approach, we focus on the classification of network traffic data. We are thus addressing the issue of applying QVC and the new stacked QVCs to big data settings. Additionally, we compare the novel idea with established approaches and also compare the results to a classical DNN model, which is motivated by the work of Dutta et al. (2020).

For our experiments, we focus on the IoT-23 (Garcia et al. 2020) dataset that contains labeled network flow data from different IoT environments. Since LSTMs are still difficult to implement as a quantum model, we use a simplified version of the model by Dutta et al., which only uses a deep neural network (DNN). Our results show, that using this kind of approach, large models can be split efficiently across different quantum computers without a significant decrease in prediction quality.

## 2 Related work

The realm of cybersecurity starts to attract researchers from QML. Gong et al. compare VQCs against DNN, support vector machines, K-nearest neighbors, Naive Bayes, and decision trees on the KDD Cup99 dataset, which focuses on network intrusion. They find that their VQC approach achieves overall the highest precision, recall, and F1 score and the lowest false negative rate compared to all other approaches. Furthermore, they test their model, which was trained with TensorFlow quantum (TFQ), with 100 randomly sampled data points on IBM quantum computers. Using the erroneous quantum hardware, they measure a prediction error of 11.96%, compared to the TFQ simulator. Thus, their approach suffers from the system noise of the quantum computer (Gong et al. 2022).

A different approach to cyber-attack detection was taken in Islam et al. (2022). The authors studied the controller area network (CAN) attack dataset for in-vehicle cyber-attack detection. They specifically focused on amplitude shift attacks, where the amplitude of a feature is shifted by a random value. Within their work, they encode the CAN data as an $13 \times 13$ image, which is used for further processing. The authors compare an LSTM, pure QVC, and a hybrid QVC. For the hybrid QVC, they used a convolutional neural network (CNN) for feature extraction. The extracted features are then used in the QVC. Their results show that the quantum-hybrid approach outperformed the other two models with an evaluation accuracy of 93.97%, which is about 6% more than the LSTM and more than 30% better than the pure quantum model.

The following three articles study QVC approaches in the realm of Botnet DGA attack detection. In Suryotrisongko and Musashi (2022a), different architectures of hybrid QVC models are compared to a classical NN on a dataset with more than 1 million domain names. For the comparisons, the same classical architecture is used in both cases, while in the quantum-hybrid model, a QVC layer is added before the output layer. Due to prolonged runtime of the quantum models, the authors sub-sampled the dataset with 100, 1000, and 10,000 samples and trained the models on the smaller datasets. The results show that both quantum and classical models achieve similar accuracy, while the quantum models' performance depends greatly on the choice of the initial random seed value. Furthermore, the authors compare the different QVC models using the noise models of IBM quantum computers.

The research is continued in Suryotrisongko and Musashi (2022b) where they compare different combinations of feature encoding, variational forms, and optimizers for QVC model training on the same dataset as in Suryotrisongko and Musashi (2022a). In total, 192 experiments are performed, and the results compared. They conclude that the TwoLocal variational model and the RawFeatureVector encoding achieve on average the best results. However, their QVC model only used one layer and two qubits, which is very small and not very expressive (Schuld et al. 2021). Similar to the previous paper, the authors evaluate a collection of models using noise models from IBM quantum computers. The results show that the selection of encoding strategy and variational form has a large influence on the results of the model, similar to model selection in classical NNs.

The last paper concerns Botnet DGA detection by a robust hybrid QVC model, which is trained using adversarial training (Suryotrisongko et al. 2022). The same dataset as in the previous papers is used. To show the vulnerability of QVC

models against adversarial attacks, first, a baseline QVC model is trained, and adversarial attacks using the fast gradient sign method (FGSM), projected gradient descent (PGD), and basic iterative method (BIM) are performed, showing declining prediction accuracy in the adversarial example attack scenarios. Using adversarial training, the prediction accuracy was improved, revealing that adversarial training can be used in QVC models to reduce the effect of adversarial example attacks. For all experiments, quantum simulators with noise models of different IBM quantum computers were used.

The literature review shows that cybersecurity is a promising research field for QML, since QVC approaches exhibit similar or improved performance compared to classical models. Furthermore, it highlights that the models' performance depends on the model selection, as seen in Suryotrisongko and Musashi (2022b). On a practical basis, it can be observed that training QVCs on real quantum hardware is not feasible due to long runtimes and noise. Using simulators, however, does not remedy the runtime issue. Both approaches have severe limitations on the number of qubits used and thus restrict the size of datasets used. Thus, utilizing smaller models in parallel can help to reduce the runtime and allow solving larger datasets.

## 3 Methods

Before we present the experiments and results, we will first provide the background of QVC in light of classical ML. Furthermore, we will introduce the concept of *vertical layers* in QVC to improve the impressibility of QML models. An introduction to the basic elements of QC can be found, for example, in Schuld and Petruccione (2021).

### 3.1 ML perspective on quantum variational circuits

Following the usual supervised learning setting for a prediction problem, we have a labeled dataset $D = \{x_i, y_i\}_{i=1}^N$ with $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{N}$ and $N \in \mathbb{N}$ samples. The goal is to find a function $f^\star \colon \mathbb{R}^n \to \mathbb{N}$ that minimizes

$$\sum_{i=1}^{N} l(y_i, \hat{y}_i), \tag{1}$$

the distance between the observed data and the prediction given by $\hat{y}_i = f^\star(x_i)$, where the difference is measured by a loss function $l : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$. The specific choice of the loss function depends on the problem at hand. For example, the binary-cross-entropy $l(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$ is the de facto standard choice for a binary classification problem.

To determine a function that minimizes the loss, we use a *model* $f_\theta$, which is a function parameterized by a vector $\theta \in \mathbb{R}^d$ and $f^\star = f_{\theta^\star}$ is obtained by solving

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} l(y_i, f_\theta(x_i)). \tag{2}$$

In the classical setting, the model is usually a NN or linear model.

In the QML setting, the model is given by

$$f_\theta(x) = \left[ \langle 0 | U^\dagger(x, \theta) M_i U(x, \theta) | 0 \rangle \right]_{i=1}^{q}, \tag{3}$$

where $\{M_i\}_{i=1}^q$ is a set of observables and $U(x, \theta)$ a parameterized unitary, called QVC. The common definition for the QVC is $U(x, \theta) = U_{enc}(x)U(\theta_1) \dots U_{enc}(x)U(\theta_d)$, which is a composition of a data encoding and a trainable quantum circuit that are repeated for several times. Each block of data encoding and trainable circuit is called a *layer*, as an analogy to hidden layers in NN. Finding a $U(x, \theta)$ that can solve the problem at hand is a challenging task and can be compared to defining the structure of a NN (Elsken et al. 2019; Pham et al. 2018). To answer this question, multiple works have been performed on possible architectures for data encoding and trainable circuits (Suzuki et al. 2020; Sim et al. 2019; Hubregtsen et al. 2021). The circuit used in this work can be found in Fig. 1.

The training process of QVCs follows the classical gradient-based approach. The gradient of a QVC with respect

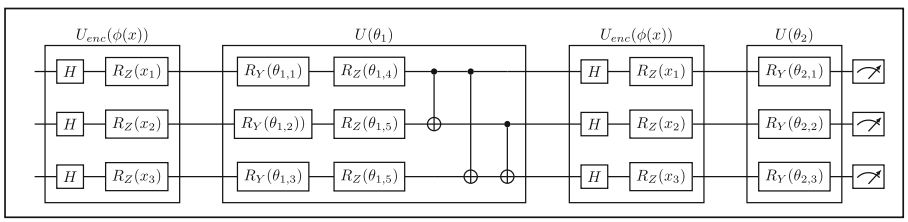


**Fig. 1** The quantum variational circuit use in this work depicted with two horizontal layers on three qubits

to a parameter can be computed using the parameter-shift rule (Mitarai et al. 2018). The gradient of $\theta_i$ is given by

$$\frac{\partial f_\theta}{\partial \theta_i} = \frac{f_{\theta_{i+}}(x) - f_{\theta_{i-}}(x)}{2}, \tag{4}$$

where $\theta_{i\pm}$ denotes a shift of the $i$-th parameter by $\pm\frac{\pi}{2}$. Thus, evaluating the QVC twice with a shift in the respective parameter results in the exact gradient. The above holds in the case of Pauli-Gates. A general version of the parameter-shift rule can be found in Wierichs et al. (2022).

This already suffices to train a QVC. The training process is depicted in Fig. 2. However, usually two more steps are integrated into the QVC. The first is a feature transformation, before the features are encoded into the QVC, and the second is a post-processing step of the measured expectation values. The former improves the training by decoupling the feature dimension of the dataset and the number of qubits in the QVC, while the latter helps to better utilize the measured expectation values for the final prediction.

For the feature encoding, a transformation $\phi(\cdot)$ is applied to the raw input values to extract only relevant features for the QVC. A common approach is to use an encoding for $\phi$, which results in a *quantum-hybrid* model. Some examples can be found in Suzuki et al. (2020); Schuld and Killoran (2019); Havlíček et al. (2019). The benefit of the feature transformation can easily be seen in the case of an angle embedding (Schuld and Petruccione 2021), where each feature is encoded by a rotation on a specific qubit. Thus, the number of qubits scales with the number of features and high-dimensional datasets become intractable to compute with today's quantum hardware or simulators. A simple and yet powerful choice for $\phi$ is a single fully connected NN layer, which will be later used within the experiments.

The final step is post-processing the measured results from the QVC to the final prediction. If more than one observable was measured in Eq. 3, but only one value is needed or vice versa, a final transformation $\psi(\cdot)$ is needed. Another case where $\psi$ is required is when the output needs to be transformed into the correct range, e.g., applying the sigmoid function for binary classification. Similar to the feature encoding, a fully connected NN layer can be used if the output from the QVC does not match the size of the final value (Schuld et al. 2020).

### 3.2 Vertical layers in QVC

The above sets the background and notation for QVCs. Next, we will introduce the idea to split the QVC part of the hybrid architecture into a collection of smaller QVCs, which we call *vertical layers*. Using parallel computations in QC is not a new idea (Niu and Todri-Sanial 2022). It is already used to speed up computations by either splitting the computation into smaller problems, which can be computed in parallel Niu et al. 2022; Schade et al. 2022; Barratt et al. 2021), or computing multiple problems in parallel (Resch et al. 2021; Mineh and Montanaro 2022).

The idea is to align more than on QVC vertically in the model. The new model is then given by

$$f_\theta(x) = \psi(f_{\theta^1}(\phi_1(x)), \ldots, f_{\theta^m}(\phi_m(x))), \tag{5}$$

which are $m$ independent QVCs with their feature transformation $\phi_i$ and a shared final transformation that combines all results into one prediction. The idea of the model is shown in Fig. 4. An analogy to this can be found in classical NNs, where multiple layers can be used in parallel. To not confuse the layers of a single QVC with the parallel layers of the full model, we will call the VQC layers *horizontal layers*.

Using layers in parallel provides multiple benefits. Instead of one large model with many qubits, parallel layers allow using multiple smaller QVCs that can be computed independently. Thus, already medium-sized hardware can compute larger models, and when using a simulator, only smaller quantum systems need to be computed, which will accelerate the process, without deteriorating the results. However, if a large quantum processor is available, the whole model can as well be computed on one chip. An updated version of the training process can be seen in Fig. 3.

Importantly, the classical encoding and output layers remain the same. Thus, no additional operations need to

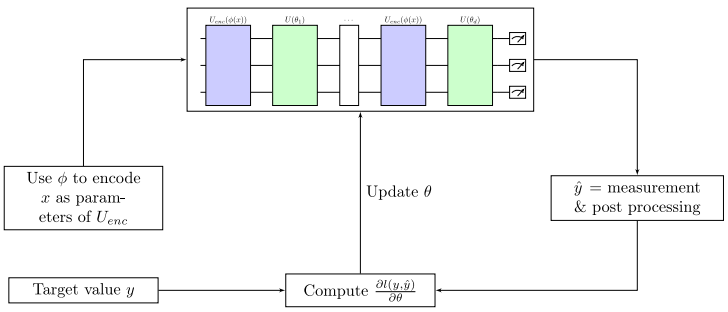


**Fig. 2** Schematic representation of the interaction between quantum and classical computer for the hybrid training of quantum variational circuits

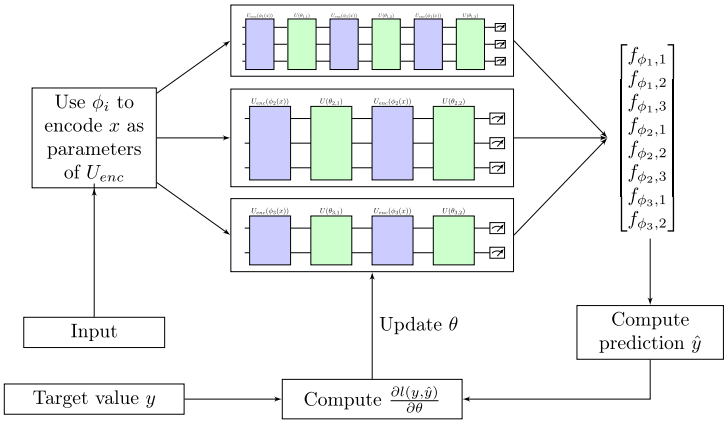


**Fig. 3** Schematic representation of the interaction between quantum and classical computer for the hybrid training of quantum variational circuits with vertical layers. In the above representation, 3 vertical layers with 3, 3, and 2 qubits each are used

be performed compared to the established setting. The intuition behind vertical layers is to compute independent feature transformations with each QVC and use the new features for the final predictions. In particular, each sub-circuit only needs to be evaluated once. If we were to use cutting and knitting for the same purpose, the number of sub-circuits would increase linearly in the number of cuts, while the number of evaluations would increase exponentially.

## 4 Experiments

To analyze the effect of vertical layers on prediction accuracy and circuit size, we first computed baseline models without vertical layers. For the models with vertical layers, we used the hyperparameter optimization framework Optuna (Akiba et al. 2019) to identify the model architecture that achieves the best accuracy. Furthermore, we repeated the baseline model computation without a classical encoding layer for the feature encoding to analyze the effect of the encoding layer.

### 4.1 Baseline models

We computed the baseline by training a classical QVC without vertical layers with the numbers of qubits ranging from 1 to 11 (the number of features in the dataset). Furthermore, the number of layers ranges from 1 to $\max(\#qubits, 7)$. This results in a total of 56 training runs. Each training was performed for 10 epochs to limit the runtime to a reasonable amount. To counter the short training, we repeated the 56 training runs with four different learning rates (LR) $10^{-3}$, $10^{-2}$, $10^{-1}$, and $2 \times 10^{-1}$.

We did not optimize the hyperparameters of the models to achieve the highest performance, since the aim of the experiments is to compare the different approaches with similar configurations. We repeated the training with different learning rates to ensure that the results were not biased due to a flawed hyperparameter choice.

The full results are shown in Tables 10 to 13 in the appendix. Here, we focus on the models with the best performance, where the performance is measured by the accuracy, precision, recall, and F1 score from the evaluation of the model. The configuration and results of the six best-performing models from the experiments are shown in Table 1.

To achieve a fair comparison, we used Optuna to find a classical model with maximum performance. The optimization was again performed with the same learning rates. Each optimization was performed for 200 trials, and the DNN configuration space consists of 1 to 5 hidden layers with 2 to 45

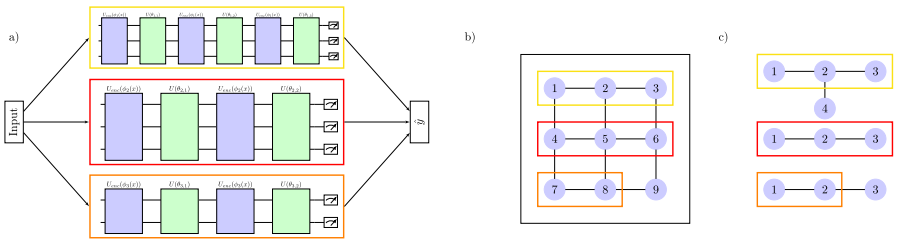


**Fig. 4** Schematic representation of a model with vertical layers. **a** Visualizes the concept of the vertical layers, **b** centralized computation of the QVCs on one processor, **c** distributing the single QVC to different quantum computers

**Table 1** Summary of the 6 best-performing quantum models and the best classical model from the optimization with Optuna

| Qubits | Layers | Parameters | LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| Classical model | | 2,077 | 0.100 | 0.9931 | 0.9931 | 0.9932 | 0.9929 |
| 9 | 6 | 217 | 0.100 | 0.9893 | 0.9892 | 0.9895 | 0.9893 |
| 7 | 5 | 155 | 0.200 | 0.9884 | 0.9888 | 0.9880 | 0.9884 |
| 8 | 3 | 145 | 0.200 | 0.9879 | 0.9876 | 0.9881 | 0.9879 |
| 11 | 5 | 243 | 0.200 | 0.9868 | 0.9872 | 0.9864 | 0.9868 |
| 10 | 6 | 241 | 0.100 | 0.9868 | 0.9865 | 0.9871 | 0.9868 |
| 7 | 7 | 183 | 0.100 | 0.9863 | 0.9863 | 0.9861 | 0.9862 |

units and sigmoid activation function. Similar to the quantum models, the classical DNN was trained with the four different learning rates, which resulted in four optimal models. Overall, the best model achieved an accuracy of 0.9931 and with two hidden layers with 40 and 38 units. The model was trained with a learning rate of $10^{-1}$. All models, quantum or classical, used ADAM (Kingma and Ba 2014) for optimization.

### 4.2 Quantum models with vertical layers

Next, the optimized QVC with vertical layers was trained. To determine the best-performing model, we again used Optuna. The model can have up to 5 vertical layers, each using up to 7 qubits, and between 2 and 4 horizontal layers. To test whether vertical layers boost the performance of the classical QVC, we limited each QVC in the model to suboptimal sizes. Similar to the baseline experiments, we performed the optimization four times with the learning rate set to $10^{-3}$, $10^{-2}$, $10^{-1}$, or $2 \times 10^{-1}$. For each learning rate, we allowed Optuna to perform 90 optimization steps.

The best model for each learning rate is shown in Table 2.

### 4.3 Feature reduction

The previous results have shown that using vertical layers is an effective method to reduce the QVCs' size. Usually, this is achieved using feature reduction techniques as a preprocessing step during the data preparation. A common method for feature reduction is the PCA, as used in Wu et al. (2021); Chen et al. (2021). To test whether vertical layers can be an alternative to feature reduction techniques, we repeat the experiments from the baseline models with a reduced feature set.

For feature reduction, we decided to use an adapted version of the Minimum Redundancy Maximum Relevance (Jo et al. 2019) algorithm. We rely solely on the maximum relevance, measured by Spearman's correlation coefficient between each feature and the label. If the absolute value of the correlation coefficient is below a certain threshold, the feature is removed from the dataset. For the experiments, we used a threshold of 0.01, which resulted in a reduction of 3 features. Compared to the PCA, the results are easy to interpret, and the features that have been removed from the dataset are "duration," "obytes," and "iipbytes."

Furthermore, duplicates in the dataset have been removed, and the dataset was balanced by undersampling the majority class. This resulted in a smaller dataset. To counter the reduced training samples, the batch size was changed to 420 to train for the same number of batches in each epoch as in the previous experiments.

With the reduced dataset, we performed the same experiments as in Sect. 4.1. However, we omitted the training with the learning rate of $10^{-3}$ since the previous results have indicated that this learning rate is too small to provide competitive performance. The full results are listed in Tables 14 to 16. The best 6 performing quantum models are listed in Table 3. Additionally, the best classical model is listed as a reference. This model was obtained from Optuna with a learning rate of $10^{-1}$. It uses two hidden layers with 23 and 40 units.

**Table 2** Summary of the best-performing quantum models using vertical layer from the optimization with Optuna

| LR | Qubits | Layers | Parameters | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| 0.200 | 7,6,4 | 3,2,3 | 295 | 0.9882 | 0.9881 | 0.9881 | 0.9881 |
| 0.100 | 7,5,3,1 | 3,4,2,3 | 293 | 0.9889 | 0.9888 | 0.9888 | 0.9889 |
| 0.010 | 7,5,4 | 4,3,4 | 311 | 0.9873 | 0.9872 | 0.9872 | 0.9872 |
| 0.001 | 7,7,5,3 | 4,3,2,3 | 419 | 0.9642 | 0.9638 | 0.9616 | 0.9686 |

**Table 3** Summary of the 6 best-performing quantum models and the best classical model from the optimization with Optuna for the reduced dataset

| Qubits | Layers | Parameters | LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| Classical model | | 1,208 | 0.10 | 0.9937 | 0.9937 | 0.9935 | 0.9940 |
| 7 | 6 | 148 | 0.20 | 0.9874 | 0.9880 | 0.9867 | 0.9873 |
| 5 | 3 | 76 | 0.20 | 0.9866 | 0.9868 | 0.9864 | 0.9866 |
| 5 | 5 | 96 | 0.10 | 0.9856 | 0.9856 | 0.9855 | 0.9855 |
| 6 | 4 | 103 | 0.20 | 0.9854 | 0.9863 | 0.9846 | 0.9853 |
| 5 | 5 | 96 | 0.20 | 0.9829 | 0.9835 | 0.9823 | 0.9828 |
| 7 | 6 | 148 | 0.10 | 0.9828 | 0.9839 | 0.9818 | 0.9827 |

## 5 Discussion

The results from Table 1 indicate that the classical model outperformed the quantum models. However, the difference is less than 0.4% between the classical model and the quantum model with 9 qubits and 6 layers. Therefore, the classical and the quantum models show about the same performance. It is interesting to note that all best-performing quantum models have been trained with a learning rate of $10^{-1}$ or $2 \times 10^{-1}$, while the classical model achieved the best performance with a learning rate of $10^{-1}$.

The classical model needs around 6 times more trainable parameters compared to the quantum models. This is not a new result. However, it confirms already known results (Chen et al. 2020; Griol-Barres et al. 2021) and, thus, validates the results of our experiments. It is likely that the quantum models can be trained with an increased learning rate, since they have fewer parameters to adjust.

Turning to the models with vertical layers, the models with learning rate $10^{-1}$ and $2 \times 10^{-1}$ show similar performance and almost the same number of trainable parameters, although they have different architectures. Interestingly, the models show slightly reduced performance compared to the baseline models. Although, the difference is only around 0.04%. This is much less than the difference between the classical model and the baseline.

The models with vertical layers need more trainable parameters compared to the baseline. On average, the use of vertical layers increases the number of parameters by 130. However, this is only true if the absolute model sizes are compared. If only the largest QVC of each model is used for comparison and omitting the classical layers, the model size is halved on average. Thus, using vertical layers effectively reduces the size of each sub-model. This is a potential benefit of models with vertical layers for big data applications in the NISQ era since multiple smaller models can be run in parallel on different quantum computers. Thus, larger datasets can be processed with less demand on the computational resources of the quantum hardware.

Albeit the potential benefits, the vertical layers resulted in a slightly reduced performance. We assume that the performance of the models with vertical layers could be improved by using more horizontal layers. Due to excessive runtime, we limited the horizontal layers to 4 during the optimization. However, the baseline results indicate that additional layers could improve the results.

Finally, comparing the results from the reduced dataset in Table 3 to both the baseline and the models with vertical layers, we observe that it shows a slight decrease in performance. The difference is about 0.19% to the baseline model and 0.15% to the model with vertical layers. Thus, vertical layers are a competitive technique to reduce the computational requirements of the QVC, without preprocessing the data. Furthermore, if comparing the size of the largest QVC in the vertical layers and the best QVC from the reduced dataset, they use the same number of qubits. However, the model for the reduced dataset uses 3 additional horizontal layers and thus is twice the size of the vertical layer model.

To ensure that the reduced performance on the smaller dataset is not due to loss of information, we can compare the results of the classical models. Here, the model on the reduced dataset shows improved performance, compared to the classical baseline model. This indicates that the reduced dataset still contains all information needed to solve the prediction problem.

We also performed additional experiments on three other datasets to evaluate if the above results also generalize to other datasets. For a report on datasets, results, and an initial discussion, see Appendix A.

Through comparison of the results, a more complex picture of the situation can be observed. Before discussing some of the differences, it should be noted that experimental settings were chosen based on the requirements of the IoT-23 dataset and left unchanged for better comparison, as it was the main object of investigation in this paper. Thus, it should not be surprising that results are a slightly worse. By adjusting the settings for each dataset, it is likely and plausible that better results can be achieved for each. This can be seen in particular, when comparing dataset and batch sizes: Per epoch, there are more than 1000 gradient updates for the IoT-23, while all others only have 5 or 6.

Overall, it can be seen that no general statement about sizes of QVCs and number of vertical layers can be made across different datasets. Thus, individual tuning will still need to be performed. Nevertheless, the same tendencies can be observed, showing that QVCs can be an alternative to classical neural networks at little loss in performance. Likewise, in particular, in cases of small available hardware, splitting QVCs into smaller vertical layers proofs to be a good option.

## 6 Conclusion

To summarize, we argued that QVCs are a promising approach to QML. However, the limited computational power of current quantum hardware is a challenge for QVC in big data scenarios like network flow prediction. To reduce the computational demand on the hardware, we proposed a novel approach, which stacks multiple smaller QVC vertically to a larger model.

We discussed the new approach in light of ensemble learning, parallel layers in a deep learning architecture, and cutting and knitting of quantum circuits. While ensemble learning combines different machine learning models to a single new model, we argue that vertical layers have more commonalities with parallel layers in classical NN. When changing the perspective from classical machine learning language to quantum computing, vertical layers follow a similar approach to circuit cutting and knitting. It turns a monolithic model into smaller subsystems that can be computed with fewer computational resources. Our approach achieves the same while omitting the additional complexity incurred by circuit knitting.

Within our experiments, we have first shown that QVC without vertical layers can solve the task of cyber-attack detection from network flow data and achieves a similar performance to a classical NN. However, the QVC needs a larger number of qubits, which limits this approach to datasets with a smaller number of features per data point. Nevertheless, one characteristic of big data is that usually numerous features are present. By computing the optimal QVC with vertical layers, we have shown that with smaller models (fewer qubits and horizontal layers), similar performance can be reached. With the current approach of scaling quantum hardware to utilize multiple smaller processes in parallel, we argue that computing multiple smaller models in parallel will provide a performance boost compared to large monolithic QVC.

To further validate our claim, we compared the QVC with vertical layers to feature reduction, a different method to decrease the computational requirements of machine learning models. Our results indicate that the classical QVC with only horizontal layers does not achieve the same performance on the reduced data compared to the full dataset. Thus, introducing vertical layers achieves the same objective, reducing the computational requirements, without deteriorating the performance.

The present paper has a clear focus on the IoT-23 dataset. While we also provide some comparative results for other datasets, more work is needed to validate the results in a more general setting. In summary, this work provides a starting point for exploring a new approach for resource-efficient QML big data settings, by utilizing multiple smaller models instead of following the current trend of monolithic models. We assume that using smaller quantum processor and models will also help to reduce errors during computation, since smaller models involve fewer gates and thus use shallow circuits.

## Appendix A. Further experiments

To illustrate the working of vertical layers better, we conducted the same experiments on three additional datasets with a similar number of attributes. The additional experiments should help to determine whether the improved performance of the vertical layers from Sect. 4.2 generalizes to other datasets.

For each of the three datasets, the following adjustments are implemented: Observations are removed to achieve an approximate balance of 50%, ensuring that the number of observations is nearly equal across categories. The datasets are partitioned into training and test sets with an approximate ratio of 65 to 35%, while ensuring that the number of data points in both sets corresponds to a multiple of the batch size. Additionally, both datasets (train and test) are balanced or nearly balanced at 50%. In addition, feature transformations, as detailed for each dataset below, are being applied to obtain about the same number of features as in the IoT-23 dataset.

### A.1 Experiments

We performed the experiments with the same setting as in Sects. 4.1 and 4.2. Detailed results and dataset descriptions are provided in the following sections.

#### A.1.1 Heart disease

The heart disease datasets (Janosi et al. 1989) contain 13 features to predict if a patient has a heart disease or not. Three categorical features are removed as they contain three or four categories, which would lead to an excessive increase in dimensionality when applying one-hot encoding, leading to 10 features. The 192 training points are divided into batches of size 32. The results for the best classical DNN and the simple CQVs are given in Table 4. The results from the Optuna experiments for the QVCs with vertical layers are provided in Table 5.

**Table 4** Summary of the 6 best-performing quantum models and the best classical model from the optimization with Optuna for the heart disease dataset

| Qubits | Layers | Parameters | LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| Classical model | | 217 | 0.200 | 0.854 | 0.855 | 0.854 | 0.856 |
| 6 | 1 | 79 | 0.200 | 0.75 | 0.748 | 0.748 | 0.749 |
| 10 | 3 | 171 | 0.200 | 0.740 | 0.737 | 0.737 | 0.739 |
| 7 | 4 | 134 | 0.100 | 0.740 | 0.735 | 0.734 | 0.744 |
| 3 | 1 | 40 | 0.010 | 0.729 | 0.726 | 0.725 | 0.729 |
| 8 | 2 | 121 | 0.100 | 0.719 | 0.718 | 0.725 | 0.730 |
| 9 | 1 | 118 | 0.200 | 0.719 | 0.719 | 0.722 | 0.723 |

**Table 5** Summary of the best-performing quantum models using vertical layer from the optimization with Optuna for the Heart disease dataset

| LR | Qubits | Layers | Parameters | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| 0.100 | 6,7,1,1,7 | 4,2,3,4,4 | 389 | 0.781 | 0.786 | 0.781 | 0.785 |
| 0.200 | 3,6,4,2 | 3,3,3,4 | 260 | 0.771 | 0.777 | 0.771 | 0.775 |
| 0.010 | 5 | 2 | 76 | 0.719 | 0.719 | 0.718 | 0.72 |
| 0.001 | 5 | 3 | 86 | 0.708 | 0.707 | 0.707 | 0.707 |

### A.1.2 Breast cancer

The breast cancer datasets (Wolberg et al. 1993) contain 30 features derived during clinical examinations of potential breast cancer material. The three-dimensional representation of each feature is transformed into a single dimension using linear discriminant analysis, thereby reducing the dataset's dimensionality to one-third of its original size. The features are used to predict if the sample is cancerous or not. Table 6 shows the results for the best classical DNN and the simple CQVs trained on 320 samples in batches of 64 and evaluated on 192 samples. The best results from the Optuna experiments for the QVCs with vertical layers are listed in Table 7.

### A.1.3 Adult

The adult dataset (Becker and Kohavi (1996)) is based on census data to predict if a person's income exceeds $ 50,000 per year. The prediction is based on 14 features in 1280 training samples, which are split into batches of size 256. Seven non-informative features are eliminated to reduce the dimensionality of the Adult dataset, and one-hot encoding is applied to the "relationship" feature leading to 12 features. The results for the best classical DNN and the simple CQVs are given in Table 8. The results from the experiments for the QVCs with vertical layers are given in 9.

**Table 6** Summary of the 6 best-performing quantum models and the best classical model from the optimization with Optuna for the breast cancer dataset

| Qubits | Layers | Parameters | LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| Classical model | | 1,227 | 0.01 | 0.979 | 0.983 | 0.978 | 0.975 |
| 4 | 4 | 77 | 0.200 | 0.969 | 0.975 | 0.967 | 0.963 |
| 7 | 4 | 134 | 0.200 | 0.964 | 0.962 | 0.958 | 0.968 |
| 2 | 1 | 27 | 0.100 | 0.953 | 0.951 | 0.947 | 0.957 |
| 10 | 6 | 231 | 0.100 | 0.953 | 0.951 | 0.947 | 0.957 |
| 7 | 1 | 40 | 0.200 | 0.953 | 0.951 | 0.947 | 0.957 |
| 3 | 1 | 92 | 0.100 | 0.953 | 0.952 | 0.949 | 0.954 |

**Table 7** Summary of the best-performing quantum models using vertical layer from the optimization with Optuna for the breast cancer dataset

| LR | Qubits | Layers | Parameters | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| 0.200 | 4,6,3 | 4,3,3 | 230 | 0.953 | 0.957 | 0.951 | 0.947 |
| 0.100 | 7,6,6 | 3,2,2 | 300 | 0.958 | 0.964 | 0.957 | 0.952 |
| 0.010 | 7 | 2 | 106 | 0.896 | 0.914 | 0.889 | 0.879 |
| 0.001 | 7,3 | 4,3 | 185 | 0.885 | 0.881 | 0.883 | 0.884 |

**Table 8** Summary of the 6 best-performing quantum models and the best classical model from the optimization with Optuna for the adult dataset

| Qubits | Layers | Parameters | LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| Classical model | | 5,359 | 0.01 | 0.818 | 0.825 | 0.817 | 0.818 |
| 4 | 2 | 69 | 0.100 | 0.786 | 0.784 | 0.786 | 0.8 |
| 11 | 1 | 166 | 0.200 | 0.788 | 0.787 | 0.788 | 0.79 |
| 8 | 1 | 121 | 0.100 | 0.786 | 0.786 | 0.786 | 0.787 |
| 10 | 2 | 171 | 0.100 | 0.773 | 0.773 | 0.773 | 0.774 |
| 11 | 3 | 210 | 0.200 | 0.766 | 0.766 | 0.766 | 0.766 |
| 9 | 2 | 154 | 0.100 | 0.764 | 0.764 | 0.764 | 0.764 |

**Table 9** Summary of the best-performing quantum models using vertical layer from the optimization with Optuna for the adult dataset

| LR | Qubits | Layers | Parameters | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|---|---|
| 0.100 | 7,2,2 | 4,4,2 | 224 | 0.771 | 0.771 | 0.771 | 0.771 |
| 0.200 | 3,7,5 | 4,4,3 | 306 | 0.762 | 0.768 | 0.76 | 0.762 |
| 0.010 | 5,6 | 3,3 | 210 | 0.734 | 0.754 | 0.729 | 0.734 |
| 0.001 | 4,2 | 4,4 | 127 | 0.639 | 0.639 | 0.639 | 0.639 |

### A.2 Discussion

Across the datasets, we can observe that the pure QVCs show performance differences from 0.01 up to 0.104 to their classical counterparts. Unsurprisingly, given the small number of training steps, the larger learning rates again deliver mostly the best results.

For the adult and breast cancer datasets, we can observe that again the gap between the approach with vertical layers and those without is about the same as the gap from classical to the best QVC-based model. For the heart disease dataset, we can observe that the best vertical layer-based approach is indeed better than the best single QVC approach. However, we can also observe that this requires a significant increase in vertical and horizontal layers. Nevertheless, a correspondingly sized model with a single vertical layer would have been significantly too large to simulate. Thus, this demonstrates the advantage of multiple smaller layers.

Overall, the number of parameters shows the expected behavior, in that the models with vertical layers have more parameters than those with a single layer, which in turn use fewer parameters than the classical models. The only exception again is the heart disease dataset, which uses a surprisingly small classical model. However, since that is also the smallest dataset, some of these deviating results might be explained by variance in the result given the small number of training steps.

## Appendix B Full experimental results

### B.1 Experiments on the full network anomaly dataset

The full results for the full IoT-23 dataset can be found in Tables 10, 11, 12, and 13.

**Table 10** Full experimental results for with learning rate $10^{-3}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.8895 | 0.8915 | 0.8869 | 0.8884 |
| 2 | 1 | 0.8886 | 0.8904 | 0.8862 | 0.8876 |
| 2 | 2 | 0.8899 | 0.8922 | 0.8873 | 0.8888 |
| 3 | 1 | 0.8891 | 0.8910 | 0.8866 | 0.8881 |
| 3 | 2 | 0.8930 | 0.8961 | 0.8900 | 0.8918 |
| 3 | 3 | 0.8925 | 0.8955 | 0.8896 | 0.8913 |
| 4 | 1 | 0.8886 | 0.8903 | 0.8862 | 0.8875 |
| 4 | 2 | 0.8926 | 0.8956 | 0.8896 | 0.8914 |
| 4 | 3 | 0.8992 | 0.9034 | 0.8959 | 0.8980 |
| 4 | 4 | 0.9301 | 0.9409 | 0.9253 | 0.9289 |
| 5 | 1 | 0.8884 | 0.8901 | 0.8860 | 0.8873 |
| 5 | 2 | 0.8934 | 0.8965 | 0.8904 | 0.8922 |
| 5 | 3 | 0.9199 | 0.9295 | 0.9152 | 0.9185 |
| 5 | 4 | 0.9289 | 0.9412 | 0.9238 | 0.9276 |
| 5 | 5 | 0.9015 | 0.9077 | 0.8976 | 0.9001 |

**Table 10** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 6 | 1 | 0.8888 | 0.8906 | 0.8864 | 0.8878 |
| 6 | 2 | 0.8945 | 0.8979 | 0.8915 | 0.8933 |
| 6 | 3 | 0.8946 | 0.8981 | 0.8915 | 0.8934 |
| 6 | 4 | 0.9034 | 0.9090 | 0.8997 | 0.9021 |
| 6 | 5 | 0.9007 | 0.9058 | 0.8971 | 0.8994 |
| 6 | 6 | 0.9363 | 0.9422 | 0.9328 | 0.9354 |
| 7 | 1 | 0.8886 | 0.8904 | 0.8862 | 0.8876 |
| 7 | 2 | 0.9013 | 0.9062 | 0.8978 | 0.9000 |
| 7 | 3 | 0.9189 | 0.9288 | 0.9142 | 0.9174 |
| 7 | 4 | 0.9205 | 0.9309 | 0.9157 | 0.9191 |
| 7 | 5 | 0.9225 | 0.9262 | 0.9196 | 0.9216 |
| 7 | 6 | 0.9435 | 0.9478 | 0.9407 | 0.9429 |
| 7 | 7 | 0.9209 | 0.9307 | 0.9162 | 0.9195 |
| 8 | 1 | 0.8888 | 0.8904 | 0.8864 | 0.8878 |
| 8 | 2 | 0.9049 | 0.9110 | 0.9010 | 0.9035 |
| 8 | 3 | 0.9217 | 0.9326 | 0.9168 | 0.9202 |
| 8 | 4 | 0.9200 | 0.9289 | 0.9155 | 0.9187 |
| 8 | 5 | 0.9238 | 0.9375 | 0.9183 | 0.9222 |
| 8 | 6 | 0.9577 | 0.9633 | 0.9546 | 0.9572 |
| 8 | 7 | 0.9508 | 0.9578 | 0.9472 | 0.9501 |
| 9 | 1 | 0.8873 | 0.8887 | 0.8851 | 0.8863 |
| 9 | 2 | 0.8969 | 0.9013 | 0.8935 | 0.8956 |
| 9 | 3 | 0.9289 | 0.9412 | 0.9237 | 0.9275 |
| 9 | 4 | 0.9319 | 0.9399 | 0.9279 | 0.9309 |
| 9 | 5 | 0.9564 | 0.9621 | 0.9533 | 0.9559 |
| 9 | 6 | 0.9225 | 0.9344 | 0.9174 | 0.9210 |
| 9 | 7 | 0.9242 | 0.9269 | 0.9218 | 0.9235 |
| 10 | 1 | 0.8880 | 0.8895 | 0.8858 | 0.8870 |
| 10 | 2 | 0.9187 | 0.9288 | 0.9139 | 0.9172 |
| 10 | 3 | 0.9129 | 0.9213 | 0.9084 | 0.9114 |
| 10 | 4 | 0.9452 | 0.9503 | 0.9422 | 0.9446 |
| 10 | 5 | 0.9576 | 0.9580 | 0.9568 | 0.9573 |
| 10 | 6 | 0.9237 | 0.9375 | 0.9182 | 0.9221 |
| 10 | 7 | 0.9272 | 0.9329 | 0.9238 | 0.9263 |
| 11 | 1 | 0.8881 | 0.8896 | 0.8859 | 0.8872 |
| 11 | 2 | 0.9232 | 0.9347 | 0.9182 | 0.9218 |
| 11 | 3 | 0.9242 | 0.9369 | 0.9189 | 0.9227 |
| 11 | 4 | 0.9255 | 0.9387 | 0.9201 | 0.9240 |
| 11 | 5 | 0.9242 | 0.9281 | 0.9213 | 0.9233 |
| 11 | 6 | 0.9640 | 0.9669 | 0.9619 | 0.9636 |
| 11 | 7 | 0.9597 | 0.9617 | 0.9581 | 0.9594 |

**Table 11** Full experimental results for with learning rate $10^{-2}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.8972 | 0.9015 | 0.8938 | 0.8959 |
| 2 | 1 | 0.9097 | 0.9199 | 0.9048 | 0.9080 |
| 2 | 2 | 0.9130 | 0.9256 | 0.9076 | 0.9112 |
| 3 | 1 | 0.9108 | 0.9213 | 0.9058 | 0.9092 |
| 3 | 2 | 0.9235 | 0.9257 | 0.9214 | 0.9228 |
| 3 | 3 | 0.9584 | 0.9605 | 0.9566 | 0.9580 |
| 4 | 1 | 0.9246 | 0.9359 | 0.9197 | 0.9232 |
| 4 | 2 | 0.9591 | 0.9639 | 0.9563 | 0.9587 |
| 4 | 3 | 0.9692 | 0.9725 | 0.9671 | 0.9689 |
| 4 | 4 | 0.9643 | 0.9668 | 0.9624 | 0.9639 |
| 5 | 1 | 0.9274 | 0.9401 | 0.9221 | 0.9259 |
| 5 | 2 | 0.9582 | 0.9600 | 0.9566 | 0.9579 |
| 5 | 3 | 0.9660 | 0.9690 | 0.9639 | 0.9657 |
| 5 | 4 | 0.9774 | 0.9796 | 0.9758 | 0.9772 |
| 5 | 5 | 0.9756 | 0.9769 | 0.9744 | 0.9754 |
| 6 | 1 | 0.9287 | 0.9411 | 0.9236 | 0.9273 |
| 6 | 2 | 0.9504 | 0.9509 | 0.9496 | 0.9501 |
| 6 | 3 | 0.9648 | 0.9669 | 0.9631 | 0.9645 |
| 6 | 4 | 0.9692 | 0.9691 | 0.9690 | 0.9690 |
| 6 | 5 | 0.9765 | 0.9776 | 0.9755 | 0.9764 |
| 6 | 6 | 0.9775 | 0.9777 | 0.9770 | 0.9773 |
| 7 | 1 | 0.9274 | 0.9385 | 0.9225 | 0.9260 |
| 7 | 2 | 0.9515 | 0.9529 | 0.9500 | 0.9511 |
| 7 | 3 | 0.9593 | 0.9608 | 0.9579 | 0.9590 |
| 7 | 4 | 0.9766 | 0.9783 | 0.9752 | 0.9764 |
| 7 | 5 | 0.9779 | 0.9793 | 0.9767 | 0.9777 |
| 7 | 6 | 0.9804 | 0.9813 | 0.9795 | 0.9803 |
| 7 | 7 | 0.9791 | 0.9797 | 0.9784 | 0.9789 |
| 8 | 1 | 0.9287 | 0.9405 | 0.9237 | 0.9273 |
| 8 | 2 | 0.9638 | 0.9655 | 0.9623 | 0.9635 |
| 8 | 3 | 0.9590 | 0.9596 | 0.9581 | 0.9588 |
| 8 | 4 | 0.9799 | 0.9818 | 0.9785 | 0.9797 |
| 8 | 5 | 0.9742 | 0.9746 | 0.9737 | 0.9741 |
| 8 | 6 | 0.9806 | 0.9819 | 0.9795 | 0.9805 |
| 8 | 7 | 0.9815 | 0.9825 | 0.9805 | 0.9813 |
| 9 | 1 | 0.9277 | 0.9391 | 0.9228 | 0.9264 |
| 9 | 2 | 0.9671 | 0.9703 | 0.9649 | 0.9668 |
| 9 | 3 | 0.9663 | 0.9674 | 0.9653 | 0.9661 |
| 9 | 4 | 0.9737 | 0.9739 | 0.9733 | 0.9736 |
| 9 | 5 | 0.9778 | 0.9784 | 0.9771 | 0.9777 |
| 9 | 6 | 0.9770 | 0.9777 | 0.9762 | 0.9768 |

**Table 11** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 9 | 7 | 0.9835 | 0.9841 | 0.9829 | 0.9834 |
| 10 | 1 | 0.9313 | 0.9417 | 0.9266 | 0.9301 |
| 10 | 2 | 0.9580 | 0.9594 | 0.9566 | 0.9577 |
| 10 | 3 | 0.9647 | 0.9661 | 0.9635 | 0.9645 |
| 10 | 4 | 0.9707 | 0.9710 | 0.9701 | 0.9705 |
| 10 | 5 | 0.9802 | 0.9811 | 0.9794 | 0.9801 |
| 10 | 6 | 0.9810 | 0.9822 | 0.9800 | 0.9809 |
| 10 | 7 | 0.9815 | 0.9812 | 0.9817 | 0.9814 |
| 11 | 1 | 0.9276 | 0.9392 | 0.9226 | 0.9263 |
| 11 | 2 | 0.9667 | 0.9706 | 0.9643 | 0.9663 |
| 11 | 3 | 0.9745 | 0.9762 | 0.9732 | 0.9743 |
| 11 | 4 | 0.9808 | 0.9822 | 0.9797 | 0.9807 |
| 11 | 5 | 0.9800 | 0.9810 | 0.9791 | 0.9799 |
| 11 | 6 | 0.9808 | 0.9819 | 0.9798 | 0.9806 |
| 11 | 7 | 0.9832 | 0.9838 | 0.9826 | 0.9831 |

**Table 12** Full experimental results for with learning rate $10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.9056 | 0.9148 | 0.9008 | 0.9039 |
| 2 | 1 | 0.9130 | 0.9265 | 0.9073 | 0.9111 |
| 2 | 2 | 0.9546 | 0.9608 | 0.9514 | 0.9541 |
| 3 | 1 | 0.9435 | 0.9479 | 0.9406 | 0.9428 |
| 3 | 2 | 0.9093 | 0.9273 | 0.9027 | 0.9070 |
| 3 | 3 | 0.9701 | 0.9706 | 0.9694 | 0.9699 |
| 4 | 1 | 0.9351 | 0.9450 | 0.9305 | 0.9339 |
| 4 | 2 | 0.9641 | 0.9656 | 0.9627 | 0.9638 |
| 4 | 3 | 0.9806 | 0.9808 | 0.9802 | 0.9805 |
| 4 | 4 | 0.9831 | 0.9830 | 0.9829 | 0.9830 |
| 5 | 1 | 0.9698 | 0.9703 | 0.9691 | 0.9696 |
| 5 | 2 | 0.9666 | 0.9688 | 0.9648 | 0.9663 |
| 5 | 3 | 0.9776 | 0.9775 | 0.9774 | 0.9775 |
| 5 | 4 | 0.9802 | 0.9800 | 0.9803 | 0.9801 |
| 5 | 5 | 0.9810 | 0.9819 | 0.9802 | 0.9809 |
| 6 | 1 | 0.9505 | 0.9557 | 0.9475 | 0.9499 |
| 6 | 2 | 0.9813 | 0.9815 | 0.9810 | 0.9812 |
| 6 | 3 | 0.9709 | 0.9729 | 0.9693 | 0.9706 |
| 6 | 4 | 0.9843 | 0.9846 | 0.9839 | 0.9842 |
| 6 | 5 | 0.9788 | 0.9785 | 0.9790 | 0.9787 |
| 6 | 6 | 0.9769 | 0.9763 | 0.9776 | 0.9768 |
| 7 | 1 | 0.9643 | 0.9687 | 0.9617 | 0.9639 |
| 7 | 2 | 0.9787 | 0.9786 | 0.9786 | 0.9786 |
| 7 | 3 | 0.9757 | 0.9759 | 0.9753 | 0.9756 |
| 7 | 4 | 0.9800 | 0.9797 | 0.9802 | 0.9799 |
| 7 | 5 | 0.9856 | 0.9855 | 0.9855 | 0.9855 |

**Table 12** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 7 | 6 | 0.9826 | 0.9824 | 0.9826 | 0.9825 |
| 7 | 7 | 0.9863 | 0.9863 | 0.9861 | 0.9862 |
| 8 | 1 | 0.9677 | 0.9714 | 0.9655 | 0.9674 |
| 8 | 2 | 0.9857 | 0.9855 | 0.9859 | 0.9857 |
| 8 | 3 | 0.9802 | 0.9808 | 0.9796 | 0.9801 |
| 8 | 4 | 0.9857 | 0.9859 | 0.9854 | 0.9856 |
| 8 | 5 | 0.9843 | 0.9844 | 0.9841 | 0.9842 |
| 8 | 6 | 0.9836 | 0.9832 | 0.9840 | 0.9835 |
| 8 | 7 | 0.9806 | 0.9801 | 0.9812 | 0.9806 |
| 9 | 1 | 0.9659 | 0.9690 | 0.9638 | 0.9656 |
| 9 | 2 | 0.9808 | 0.9809 | 0.9805 | 0.9807 |
| 9 | 3 | 0.9816 | 0.9812 | 0.9818 | 0.9815 |
| 9 | 4 | 0.9837 | 0.9840 | 0.9834 | 0.9837 |
| 9 | 5 | 0.9785 | 0.9780 | 0.9788 | 0.9784 |
| 9 | 6 | 0.9893 | 0.9892 | 0.9895 | 0.9893 |
| 9 | 7 | 0.9837 | 0.9842 | 0.9832 | 0.9836 |
| 10 | 1 | 0.9632 | 0.9677 | 0.9605 | 0.9628 |
| 10 | 2 | 0.9607 | 0.9631 | 0.9589 | 0.9604 |
| 10 | 3 | 0.9844 | 0.9844 | 0.9843 | 0.9844 |
| 10 | 4 | 0.9797 | 0.9794 | 0.9799 | 0.9796 |
| 10 | 5 | 0.9823 | 0.9822 | 0.9822 | 0.9822 |
| 10 | 6 | 0.9868 | 0.9865 | 0.9871 | 0.9868 |
| 10 | 7 | 0.9797 | 0.9810 | 0.9786 | 0.9796 |
| 11 | 1 | 0.9686 | 0.9719 | 0.9665 | 0.9683 |
| 11 | 2 | 0.9674 | 0.9698 | 0.9656 | 0.9671 |
| 11 | 3 | 0.9825 | 0.9831 | 0.9818 | 0.9824 |
| 11 | 4 | 0.9798 | 0.9793 | 0.9805 | 0.9798 |
| 11 | 5 | 0.9846 | 0.9847 | 0.9844 | 0.9845 |
| 11 | 6 | 0.9842 | 0.9844 | 0.9838 | 0.9841 |
| 11 | 7 | 0.9826 | 0.9821 | 0.9831 | 0.9825 |

**Table 13** Full experimental results for with learning rate $2 * 10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.9044 | 0.9119 | 0.9001 | 0.9029 |
| 2 | 1 | 0.9126 | 0.9271 | 0.9067 | 0.9106 |
| 2 | 2 | 0.9659 | 0.9653 | 0.9669 | 0.9658 |
| 3 | 1 | 0.9552 | 0.9577 | 0.9532 | 0.9548 |
| 3 | 2 | 0.9790 | 0.9788 | 0.9790 | 0.9789 |
| 3 | 3 | 0.9731 | 0.9733 | 0.9727 | 0.9730 |
| 4 | 1 | 0.9565 | 0.9585 | 0.9548 | 0.9561 |
| 4 | 2 | 0.9651 | 0.9693 | 0.9626 | 0.9648 |
| 4 | 3 | 0.9745 | 0.9753 | 0.9736 | 0.9743 |
| 4 | 4 | 0.9823 | 0.9822 | 0.9822 | 0.9822 |
| 5 | 1 | 0.9389 | 0.9468 | 0.9349 | 0.9380 |

**Table 13** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 5 | 2 | 0.9812 | 0.9808 | 0.9817 | 0.9812 |
| 5 | 3 | 0.9841 | 0.9838 | 0.9843 | 0.9840 |
| 5 | 4 | 0.9799 | 0.9799 | 0.9797 | 0.9798 |
| 5 | 5 | 0.9651 | 0.9664 | 0.9639 | 0.9649 |
| 6 | 1 | 0.9545 | 0.9597 | 0.9515 | 0.9539 |
| 6 | 2 | 0.9760 | 0.9771 | 0.9750 | 0.9758 |
| 6 | 3 | 0.9759 | 0.9764 | 0.9752 | 0.9757 |
| 6 | 4 | 0.9760 | 0.9776 | 0.9747 | 0.9758 |
| 6 | 5 | 0.9855 | 0.9852 | 0.9857 | 0.9854 |
| 6 | 6 | 0.9835 | 0.9831 | 0.9840 | 0.9835 |
| 7 | 1 | 0.9489 | 0.9554 | 0.9455 | 0.9482 |
| 7 | 2 | 0.9790 | 0.9785 | 0.9794 | 0.9789 |
| 7 | 3 | 0.9794 | 0.9794 | 0.9792 | 0.9793 |
| 7 | 4 | 0.9861 | 0.9862 | 0.9860 | 0.9861 |
| 7 | 5 | 0.9884 | 0.9888 | 0.9880 | 0.9884 |
| 7 | 6 | 0.9771 | 0.9787 | 0.9758 | 0.9769 |
| 7 | 7 | 0.9846 | 0.9844 | 0.9848 | 0.9846 |
| 8 | 1 | 0.9308 | 0.9424 | 0.9259 | 0.9295 |
| 8 | 2 | 0.9804 | 0.9801 | 0.9806 | 0.9803 |
| 8 | 3 | 0.9879 | 0.9876 | 0.9881 | 0.9879 |
| 8 | 4 | 0.9680 | 0.9699 | 0.9664 | 0.9677 |
| 8 | 5 | 0.9731 | 0.9749 | 0.9716 | 0.9728 |
| 8 | 6 | 0.9842 | 0.9843 | 0.9840 | 0.9841 |
| 8 | 7 | 0.9807 | 0.9814 | 0.9800 | 0.9806 |
| 9 | 1 | 0.9600 | 0.9620 | 0.9584 | 0.9597 |
| 9 | 2 | 0.9805 | 0.9811 | 0.9798 | 0.9804 |
| 9 | 3 | 0.9858 | 0.9855 | 0.9860 | 0.9858 |
| 9 | 4 | 0.9721 | 0.9743 | 0.9704 | 0.9718 |
| 9 | 5 | 0.9739 | 0.9760 | 0.9723 | 0.9737 |
| 9 | 6 | 0.9797 | 0.9796 | 0.9797 | 0.9797 |
| 9 | 7 | 0.9843 | 0.9838 | 0.9847 | 0.9842 |
| 10 | 1 | 0.9676 | 0.9709 | 0.9655 | 0.9673 |
| 10 | 2 | 0.9537 | 0.9601 | 0.9504 | 0.9531 |
| 10 | 3 | 0.9802 | 0.9800 | 0.9802 | 0.9801 |
| 10 | 4 | 0.9825 | 0.9821 | 0.9828 | 0.9824 |
| 10 | 5 | 0.9799 | 0.9800 | 0.9797 | 0.9798 |
| 10 | 6 | 0.9768 | 0.9772 | 0.9762 | 0.9767 |
| 10 | 7 | 0.9826 | 0.9840 | 0.9814 | 0.9825 |
| 11 | 1 | 0.9393 | 0.9486 | 0.9350 | 0.9383 |
| 11 | 2 | 0.9839 | 0.9835 | 0.9842 | 0.9838 |
| 11 | 3 | 0.9563 | 0.9617 | 0.9533 | 0.9558 |
| 11 | 4 | 0.9762 | 0.9777 | 0.9750 | 0.9760 |
| 11 | 5 | 0.9868 | 0.9872 | 0.9864 | 0.9868 |
| 11 | 6 | 0.9838 | 0.9835 | 0.9840 | 0.9838 |
| 11 | 7 | 0.9862 | 0.9866 | 0.9857 | 0.9861 |

## B.2 Results on the reduced network anomaly dataset

The full results for the reduced IoT-23 dataset can be found in Tables 14, 15, and 16.

**Table 14** Full experimental results for with learning rate $10^{-2}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.9031 | 0.9093 | 0.8991 | 0.9017 |
| 2 | 1 | 0.9229 | 0.9321 | 0.9184 | 0.9216 |
| 2 | 2 | 0.8899 | 0.9024 | 0.8842 | 0.8876 |
| 3 | 1 | 0.9234 | 0.9332 | 0.9187 | 0.9220 |
| 3 | 2 | 0.9627 | 0.9674 | 0.9601 | 0.9623 |
| 3 | 3 | 0.9628 | 0.9675 | 0.9602 | 0.9624 |
| 4 | 1 | 0.9198 | 0.9279 | 0.9156 | 0.9185 |
| 4 | 2 | 0.9627 | 0.9667 | 0.9602 | 0.9623 |
| 4 | 3 | 0.9643 | 0.9687 | 0.9618 | 0.9639 |
| 4 | 4 | 0.9611 | 0.9625 | 0.9597 | 0.9608 |
| 5 | 1 | 0.9176 | 0.9254 | 0.9134 | 0.9163 |
| 5 | 2 | 0.9578 | 0.9593 | 0.9564 | 0.9575 |
| 5 | 3 | 0.9607 | 0.9620 | 0.9593 | 0.9604 |
| 5 | 4 | 0.9708 | 0.9712 | 0.9701 | 0.9706 |
| 5 | 5 | 0.9782 | 0.9792 | 0.9772 | 0.9780 |
| 6 | 1 | 0.9170 | 0.9248 | 0.9128 | 0.9157 |
| 6 | 2 | 0.9582 | 0.9589 | 0.9573 | 0.9579 |
| 6 | 3 | 0.9604 | 0.9609 | 0.9596 | 0.9601 |
| 6 | 4 | 0.9772 | 0.9778 | 0.9766 | 0.9771 |
| 6 | 5 | 0.9788 | 0.9794 | 0.9781 | 0.9787 |
| 6 | 6 | 0.9704 | 0.9702 | 0.9704 | 0.9703 |
| 7 | 1 | 0.9166 | 0.9242 | 0.9124 | 0.9153 |
| 7 | 2 | 0.9614 | 0.9625 | 0.9601 | 0.9611 |
| 7 | 3 | 0.9737 | 0.9741 | 0.9731 | 0.9735 |
| 7 | 4 | 0.9741 | 0.9753 | 0.9730 | 0.9739 |
| 7 | 5 | 0.9800 | 0.9808 | 0.9791 | 0.9799 |
| 7 | 6 | 0.9803 | 0.9812 | 0.9795 | 0.9802 |
| 7 | 7 | 0.9825 | 0.9823 | 0.9826 | 0.9824 |

**Table 15** Full experimental results for with learning rate $10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.8769 | 0.8767 | 0.8785 | 0.8767 |
| 2 | 1 | 0.9066 | 0.9255 | 0.8998 | 0.9041 |
| 2 | 2 | 0.9412 | 0.9411 | 0.9408 | 0.9409 |
| 3 | 1 | 0.9629 | 0.9623 | 0.9642 | 0.9628 |
| 3 | 2 | 0.9693 | 0.9710 | 0.9678 | 0.9690 |
| 3 | 3 | 0.9750 | 0.9745 | 0.9753 | 0.9749 |

**Table 15** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 4 | 1 | 0.9448 | 0.9464 | 0.9431 | 0.9443 |
| 4 | 2 | 0.9735 | 0.9732 | 0.9737 | 0.9734 |
| 4 | 3 | 0.9688 | 0.9685 | 0.9688 | 0.9687 |
| 4 | 4 | 0.9825 | 0.9837 | 0.9816 | 0.9824 |
| 5 | 1 | 0.9623 | 0.9666 | 0.9598 | 0.9619 |
| 5 | 2 | 0.9646 | 0.9650 | 0.9639 | 0.9644 |
| 5 | 3 | 0.9797 | 0.9793 | 0.9799 | 0.9796 |
| 5 | 4 | 0.9779 | 0.9790 | 0.9769 | 0.9778 |
| 5 | 5 | 0.9856 | 0.9856 | 0.9855 | 0.9855 |
| 6 | 1 | 0.9681 | 0.9710 | 0.9661 | 0.9678 |
| 6 | 2 | 0.9713 | 0.9720 | 0.9705 | 0.9711 |
| 6 | 3 | 0.9765 | 0.9763 | 0.9764 | 0.9764 |
| 6 | 4 | 0.9751 | 0.9757 | 0.9744 | 0.9750 |
| 6 | 5 | 0.9735 | 0.9742 | 0.9726 | 0.9733 |
| 6 | 6 | 0.9755 | 0.9762 | 0.9748 | 0.9754 |
| 7 | 1 | 0.9758 | 0.9781 | 0.9741 | 0.9756 |
| 7 | 2 | 0.9715 | 0.9715 | 0.9713 | 0.9714 |
| 7 | 3 | 0.9818 | 0.9835 | 0.9805 | 0.9817 |
| 7 | 4 | 0.9791 | 0.9807 | 0.9778 | 0.9790 |
| 7 | 5 | 0.9827 | 0.9828 | 0.9824 | 0.9826 |
| 7 | 6 | 0.9828 | 0.9839 | 0.9818 | 0.9827 |
| 7 | 7 | 0.9807 | 0.9814 | 0.9799 | 0.9805 |

**Table 16** Full experimental results for with learning rate $2 * 10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.8667 | 0.8685 | 0.8696 | 0.8667 |
| 2 | 1 | 0.9066 | 0.9255 | 0.8998 | 0.9041 |
| 2 | 2 | 0.9408 | 0.9413 | 0.9398 | 0.9405 |
| 3 | 1 | 0.9570 | 0.9567 | 0.9587 | 0.9569 |
| 3 | 2 | 0.9736 | 0.9731 | 0.9743 | 0.9736 |
| 3 | 3 | 0.9789 | 0.9803 | 0.9777 | 0.9787 |
| 4 | 1 | 0.8810 | 0.8945 | 0.8877 | 0.8808 |
| 4 | 2 | 0.9644 | 0.9638 | 0.9651 | 0.9643 |
| 4 | 3 | 0.9432 | 0.9443 | 0.9461 | 0.9431 |
| 4 | 4 | 0.9773 | 0.9769 | 0.9778 | 0.9773 |
| 5 | 1 | 0.9537 | 0.9536 | 0.9556 | 0.9536 |
| 5 | 2 | 0.9558 | 0.9589 | 0.9536 | 0.9554 |
| 5 | 3 | 0.9866 | 0.9868 | 0.9864 | 0.9866 |
| 5 | 4 | 0.9819 | 0.9815 | 0.9824 | 0.9819 |
| 5 | 5 | 0.9829 | 0.9835 | 0.9823 | 0.9828 |
| 6 | 1 | 0.9539 | 0.9538 | 0.9558 | 0.9538 |
| 6 | 2 | 0.9678 | 0.9713 | 0.9655 | 0.9674 |
| 6 | 3 | 0.9799 | 0.9795 | 0.9802 | 0.9798 |
| 6 | 4 | 0.9854 | 0.9863 | 0.9846 | 0.9853 |

**Table 16** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 6 | 5 | 0.9810 | 0.9806 | 0.9812 | 0.9809 |
| 6 | 6 | 0.9791 | 0.9806 | 0.9778 | 0.9789 |
| 7 | 1 | 0.9654 | 0.9674 | 0.9638 | 0.9652 |
| 7 | 2 | 0.9826 | 0.9823 | 0.9830 | 0.9826 |
| 7 | 3 | 0.9754 | 0.9756 | 0.9750 | 0.9753 |
| 7 | 4 | 0.9679 | 0.9680 | 0.9675 | 0.9677 |
| 7 | 5 | 0.9799 | 0.9794 | 0.9805 | 0.9798 |
| 7 | 6 | 0.9874 | 0.9880 | 0.9867 | 0.9873 |
| 7 | 7 | 0.9824 | 0.9831 | 0.9816 | 0.9823 |

### B.3 Experiments on the heart disease dataset

The full results for the heart disease dataset can be found in Tables 17, 18, 19, and 20.

**Table 17** Full experimental results for the heart disease dataset with learning rate $10^{-3}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.6354 | 0.6335 | 0.6333 | 0.6337 |
| 2 | 1 | 0.4792 | 0.4755 | 0.4758 | 0.4756 |
| 2 | 2 | 0.4688 | 0.3873 | 0.4935 | 0.4826 |
| 3 | 1 | 0.6875 | 0.6874 | 0.6889 | 0.6882 |
| 3 | 2 | 0.4896 | 0.4383 | 0.5105 | 0.5188 |
| 3 | 3 | 0.5104 | 0.4748 | 0.4961 | 0.4951 |
| 4 | 1 | 0.6354 | 0.6236 | 0.6268 | 0.6380 |
| 4 | 2 | 0.3542 | 0.3471 | 0.3490 | 0.3463 |
| 4 | 3 | 0.4896 | 0.4212 | 0.4699 | 0.4509 |
| 4 | 4 | 0.5104 | 0.4684 | 0.5301 | 0.5491 |
| 5 | 1 | 0.4583 | 0.4574 | 0.4627 | 0.4621 |
| 5 | 2 | 0.4583 | 0.3806 | 0.4824 | 0.4567 |
| 5 | 3 | 0.4688 | 0.4659 | 0.4752 | 0.4741 |
| 5 | 4 | 0.4792 | 0.4503 | 0.4954 | 0.4937 |
| 5 | 5 | 0.4271 | 0.4164 | 0.4203 | 0.4170 |
| 6 | 1 | 0.5000 | 0.4535 | 0.4837 | 0.4777 |
| 6 | 2 | 0.3958 | 0.3956 | 0.3961 | 0.3964 |
| 6 | 3 | 0.4896 | 0.4639 | 0.5052 | 0.5069 |
| 6 | 4 | 0.4271 | 0.4164 | 0.4203 | 0.4170 |
| 6 | 5 | 0.3958 | 0.3862 | 0.4052 | 0.3954 |
| 6 | 6 | 0.5833 | 0.5767 | 0.5778 | 0.5803 |
| 7 | 1 | 0.5938 | 0.5862 | 0.5876 | 0.5912 |
| 7 | 2 | 0.4792 | 0.4141 | 0.4601 | 0.4375 |
| 7 | 3 | 0.5313 | 0.3835 | 0.5026 | 0.5163 |
| 7 | 4 | 0.5417 | 0.4921 | 0.5242 | 0.5352 |

**Table 17** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 7 | 5 | 0.5313 | 0.5225 | 0.5248 | 0.5259 |
| 7 | 6 | 0.4583 | 0.4497 | 0.4680 | 0.4646 |
| 7 | 7 | 0.5000 | 0.4945 | 0.4954 | 0.4953 |
| 8 | 1 | 0.4375 | 0.4253 | 0.4484 | 0.4412 |
| 8 | 2 | 0.4375 | 0.3672 | 0.4601 | 0.4152 |
| 8 | 3 | 0.3958 | 0.3827 | 0.3882 | 0.3825 |
| 8 | 4 | 0.6146 | 0.6135 | 0.6137 | 0.6135 |
| 8 | 5 | 0.5417 | 0.4759 | 0.5216 | 0.5369 |
| 8 | 6 | 0.5208 | 0.5206 | 0.5242 | 0.5244 |
| 8 | 7 | 0.5104 | 0.4982 | 0.5026 | 0.5028 |
| 9 | 1 | 0.6563 | 0.6553 | 0.6556 | 0.6552 |
| 9 | 2 | 0.5313 | 0.4684 | 0.5549 | 0.6250 |
| 9 | 3 | 0.5104 | 0.3379 | 0.4804 | 0.2606 |
| 9 | 4 | 0.4583 | 0.4386 | 0.4719 | 0.4654 |
| 9 | 5 | 0.4896 | 0.4212 | 0.4699 | 0.4509 |
| 9 | 6 | 0.5938 | 0.5934 | 0.5941 | 0.5938 |
| 9 | 7 | 0.5313 | 0.4971 | 0.5170 | 0.5214 |
| 10 | 1 | 0.6563 | 0.6562 | 0.6595 | 0.6600 |
| 10 | 2 | 0.4896 | 0.3287 | 0.4608 | 0.2554 |
| 10 | 3 | 0.5729 | 0.5555 | 0.5627 | 0.5703 |
| 10 | 4 | 0.5833 | 0.5714 | 0.5752 | 0.5808 |
| 10 | 5 | 0.4271 | 0.4240 | 0.4333 | 0.4306 |
| 10 | 6 | 0.5313 | 0.5308 | 0.5353 | 0.5357 |
| 10 | 7 | 0.4688 | 0.4659 | 0.4752 | 0.4741 |
| 11 | 1 | 0.4063 | 0.3701 | 0.4229 | 0.3913 |
| 11 | 2 | 0.5313 | 0.5161 | 0.5444 | 0.5536 |
| 11 | 3 | 0.5417 | 0.5399 | 0.5477 | 0.5493 |
| 11 | 4 | 0.5833 | 0.5152 | 0.5621 | 0.6173 |
| 11 | 5 | 0.4375 | 0.4000 | 0.4235 | 0.4058 |
| 11 | 6 | 0.5729 | 0.5649 | 0.5667 | 0.5694 |
| 11 | 7 | 0.5729 | 0.5672 | 0.5680 | 0.5696 |

**Table 18** Full experimental results for the heart disease dataset with learning rate $10^{-2}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5521 | 0.4397 | 0.5261 | 0.5852 |
| 2 | 1 | 0.5938 | 0.5530 | 0.5771 | 0.6087 |
| 2 | 2 | 0.4688 | 0.4471 | 0.4582 | 0.4531 |
| 3 | 1 | 0.7292 | 0.7262 | 0.7255 | 0.7295 |
| 3 | 2 | 0.5625 | 0.5623 | 0.5660 | 0.5665 |
| 3 | 3 | 0.5104 | 0.5078 | 0.5078 | 0.5079 |
| 4 | 1 | 0.3854 | 0.3848 | 0.3850 | 0.3854 |
| 4 | 2 | 0.4688 | 0.3191 | 0.5000 | 0.2344 |
| 4 | 3 | 0.4271 | 0.4265 | 0.4268 | 0.4271 |
| 4 | 4 | 0.5521 | 0.5071 | 0.5353 | 0.5498 |

**Table 18** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 5 | 1 | 0.5521 | 0.4397 | 0.5261 | 0.5852 |
| 5 | 2 | 0.5000 | 0.4773 | 0.5150 | 0.5194 |
| 5 | 3 | 0.5833 | 0.5743 | 0.5765 | 0.5804 |
| 5 | 4 | 0.5938 | 0.5934 | 0.5941 | 0.5938 |
| 5 | 5 | 0.5729 | 0.5300 | 0.5562 | 0.5792 |
| 6 | 1 | 0.4792 | 0.4444 | 0.4654 | 0.4573 |
| 6 | 2 | 0.4167 | 0.3902 | 0.4052 | 0.3921 |
| 6 | 3 | 0.4167 | 0.3705 | 0.4013 | 0.3724 |
| 6 | 4 | 0.5313 | 0.5121 | 0.5209 | 0.5234 |
| 6 | 5 | 0.5000 | 0.4459 | 0.4824 | 0.4743 |
| 6 | 6 | 0.5313 | 0.4910 | 0.5157 | 0.5208 |
| 7 | 1 | 0.6771 | 0.6711 | 0.6882 | 0.7109 |
| 7 | 2 | 0.6146 | 0.5911 | 0.6020 | 0.6229 |
| 7 | 3 | 0.5208 | 0.4690 | 0.5033 | 0.5048 |
| 7 | 4 | 0.4688 | 0.4516 | 0.4595 | 0.4559 |
| 7 | 5 | 0.5313 | 0.4910 | 0.5157 | 0.5208 |
| 7 | 6 | 0.5417 | 0.4565 | 0.5190 | 0.5403 |
| 7 | 7 | 0.5729 | 0.5418 | 0.5588 | 0.5742 |
| 8 | 1 | 0.6875 | 0.6761 | 0.6784 | 0.6970 |
| 8 | 2 | 0.5625 | 0.4706 | 0.5386 | 0.5947 |
| 8 | 3 | 0.5417 | 0.4921 | 0.5242 | 0.5352 |
| 8 | 4 | 0.6250 | 0.6042 | 0.6131 | 0.6336 |
| 8 | 5 | 0.5417 | 0.4991 | 0.5255 | 0.5348 |
| 8 | 6 | 0.6771 | 0.6609 | 0.6660 | 0.6924 |
| 8 | 7 | 0.4792 | 0.4709 | 0.4732 | 0.4723 |
| 9 | 1 | 0.5938 | 0.5862 | 0.6046 | 0.6172 |
| 9 | 2 | 0.4375 | 0.4313 | 0.4458 | 0.4417 |
| 9 | 3 | 0.5104 | 0.4136 | 0.4869 | 0.4702 |
| 9 | 4 | 0.5729 | 0.5515 | 0.5614 | 0.5712 |
| 9 | 5 | 0.6771 | 0.6609 | 0.6660 | 0.6924 |
| 9 | 6 | 0.5833 | 0.5382 | 0.5660 | 0.5962 |
| 9 | 7 | 0.5729 | 0.5157 | 0.5536 | 0.5876 |
| 10 | 1 | 0.6042 | 0.5547 | 0.5856 | 0.6343 |
| 10 | 2 | 0.3438 | 0.2874 | 0.3627 | 0.2756 |
| 10 | 3 | 0.4479 | 0.4474 | 0.4477 | 0.4479 |
| 10 | 4 | 0.5729 | 0.5725 | 0.5732 | 0.5729 |
| 10 | 5 | 0.4792 | 0.3407 | 0.4523 | 0.3242 |
| 10 | 6 | 0.5208 | 0.3934 | 0.4941 | 0.4783 |
| 10 | 7 | 0.5833 | 0.4592 | 0.5556 | 0.7802 |
| 11 | 1 | 0.4063 | 0.3984 | 0.4007 | 0.3982 |
| 11 | 2 | 0.5313 | 0.3992 | 0.5039 | 0.5167 |
| 11 | 3 | 0.5208 | 0.5104 | 0.5137 | 0.5144 |
| 11 | 4 | 0.6563 | 0.6544 | 0.6542 | 0.6547 |
| 11 | 5 | 0.5208 | 0.4763 | 0.5046 | 0.5063 |
| 11 | 6 | 0.5938 | 0.5589 | 0.5784 | 0.6042 |
| 11 | 7 | 0.6146 | 0.5697 | 0.5967 | 0.6461 |

**Table 19** Full experimental results for the heart disease dataset with learning rate $10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5313 | 0.3469 | 0.5000 | 0.2656 |
| 2 | 1 | 0.6458 | 0.6457 | 0.6497 | 0.6507 |
| 2 | 2 | 0.5313 | 0.3469 | 0.5000 | 0.2656 |
| 3 | 1 | 0.7083 | 0.7072 | 0.7072 | 0.7072 |
| 3 | 2 | 0.6042 | 0.5674 | 0.5882 | 0.6206 |
| 3 | 3 | 0.6771 | 0.6574 | 0.6647 | 0.6985 |
| 4 | 1 | 0.4479 | 0.4464 | 0.4464 | 0.4465 |
| 4 | 2 | 0.7188 | 0.7073 | 0.7092 | 0.7344 |
| 4 | 3 | 0.6458 | 0.6329 | 0.6366 | 0.6508 |
| 4 | 4 | 0.5729 | 0.5649 | 0.5667 | 0.5694 |
| 5 | 1 | 0.6250 | 0.6042 | 0.6131 | 0.6336 |
| 5 | 2 | 0.6667 | 0.6661 | 0.6667 | 0.6661 |
| 5 | 3 | 0.6875 | 0.6825 | 0.6824 | 0.6883 |
| 5 | 4 | 0.7083 | 0.7000 | 0.7007 | 0.7157 |
| 5 | 5 | 0.6250 | 0.6209 | 0.6209 | 0.6231 |
| 6 | 1 | 0.5729 | 0.5591 | 0.5863 | 0.6040 |
| 6 | 2 | 0.6250 | 0.6113 | 0.6157 | 0.6277 |
| 6 | 3 | 0.4792 | 0.3824 | 0.4562 | 0.4069 |
| 6 | 4 | 0.6771 | 0.6768 | 0.6778 | 0.6771 |
| 6 | 5 | 0.6563 | 0.6562 | 0.6595 | 0.6600 |
| 6 | 6 | 0.5729 | 0.5649 | 0.5667 | 0.5694 |
| 7 | 1 | 0.5521 | 0.4834 | 0.5314 | 0.5563 |
| 7 | 2 | 0.7188 | 0.7187 | 0.7209 | 0.7204 |
| 7 | 3 | 0.6667 | 0.6630 | 0.6627 | 0.6656 |
| 7 | 4 | 0.7396 | 0.7347 | 0.7340 | 0.7436 |
| 7 | 5 | 0.5729 | 0.5232 | 0.5549 | 0.5829 |
| 7 | 6 | 0.5833 | 0.5714 | 0.5752 | 0.5808 |
| 7 | 7 | 0.5833 | 0.5832 | 0.5843 | 0.5840 |
| 8 | 1 | 0.5625 | 0.5383 | 0.5503 | 0.5594 |
| 8 | 2 | 0.7188 | 0.7180 | 0.7248 | 0.7304 |
| 8 | 3 | 0.6250 | 0.6248 | 0.6288 | 0.6297 |
| 8 | 4 | 0.5729 | 0.5591 | 0.5641 | 0.5697 |
| 8 | 5 | 0.5938 | 0.5934 | 0.5941 | 0.5938 |
| 8 | 6 | 0.6146 | 0.6145 | 0.6176 | 0.6180 |
| 8 | 7 | 0.6250 | 0.6235 | 0.6235 | 0.6235 |
| 9 | 1 | 0.5833 | 0.5788 | 0.5922 | 0.5991 |
| 9 | 2 | 0.6250 | 0.6113 | 0.6392 | 0.6715 |
| 9 | 3 | 0.6771 | 0.6742 | 0.6739 | 0.6759 |
| 9 | 4 | 0.7188 | 0.7135 | 0.7131 | 0.7219 |
| 9 | 5 | 0.6250 | 0.6224 | 0.6222 | 0.6231 |

**Table 20** Full experimental results for the heart disease dataset with learning rate $2 * 10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5313 | 0.3469 | 0.5000 | 0.2656 |
| 2 | 1 | 0.6458 | 0.6457 | 0.6497 | 0.6507 |
| 2 | 2 | 0.5313 | 0.3469 | 0.5000 | 0.2656 |
| 3 | 1 | 0.7083 | 0.7072 | 0.7072 | 0.7072 |
| 3 | 2 | 0.6042 | 0.5674 | 0.5882 | 0.6206 |
| 3 | 3 | 0.6771 | 0.6574 | 0.6647 | 0.6985 |
| 4 | 1 | 0.4479 | 0.4464 | 0.4464 | 0.4465 |
| 4 | 2 | 0.7188 | 0.7073 | 0.7092 | 0.7344 |
| 4 | 3 | 0.6458 | 0.6329 | 0.6366 | 0.6508 |
| 4 | 4 | 0.5729 | 0.5649 | 0.5667 | 0.5694 |
| 5 | 1 | 0.6250 | 0.6042 | 0.6131 | 0.6336 |
| 5 | 2 | 0.6667 | 0.6661 | 0.6667 | 0.6661 |
| 5 | 3 | 0.6875 | 0.6825 | 0.6824 | 0.6883 |
| 5 | 4 | 0.7083 | 0.7000 | 0.7007 | 0.7157 |
| 5 | 5 | 0.6250 | 0.6209 | 0.6209 | 0.6231 |
| 6 | 1 | 0.5729 | 0.5591 | 0.5863 | 0.6040 |
| 6 | 2 | 0.6250 | 0.6113 | 0.6157 | 0.6277 |
| 6 | 3 | 0.4792 | 0.3824 | 0.4562 | 0.4069 |
| 6 | 4 | 0.6771 | 0.6768 | 0.6778 | 0.6771 |
| 6 | 5 | 0.6563 | 0.6562 | 0.6595 | 0.6600 |
| 6 | 6 | 0.5729 | 0.5649 | 0.5667 | 0.5694 |
| 7 | 1 | 0.5521 | 0.4834 | 0.5314 | 0.5563 |
| 7 | 2 | 0.7188 | 0.7187 | 0.7209 | 0.7204 |
| 7 | 3 | 0.6667 | 0.6630 | 0.6627 | 0.6656 |
| 7 | 4 | 0.7396 | 0.7347 | 0.7340 | 0.7436 |
| 7 | 5 | 0.5729 | 0.5232 | 0.5549 | 0.5829 |
| 7 | 6 | 0.5833 | 0.5714 | 0.5752 | 0.5808 |
| 7 | 7 | 0.5833 | 0.5832 | 0.5843 | 0.5840 |
| 8 | 1 | 0.5625 | 0.5383 | 0.5503 | 0.5594 |
| 8 | 2 | 0.7188 | 0.7180 | 0.7248 | 0.7304 |
| 8 | 3 | 0.6250 | 0.6248 | 0.6288 | 0.6297 |
| 8 | 4 | 0.5729 | 0.5591 | 0.5641 | 0.5697 |
| 8 | 5 | 0.5938 | 0.5934 | 0.5941 | 0.5938 |
| 8 | 6 | 0.6146 | 0.6145 | 0.6176 | 0.6180 |
| 8 | 7 | 0.6250 | 0.6235 | 0.6235 | 0.6235 |
| 9 | 1 | 0.5833 | 0.5788 | 0.5922 | 0.5991 |
| 9 | 2 | 0.6250 | 0.6113 | 0.6392 | 0.6715 |
| 9 | 3 | 0.6771 | 0.6742 | 0.6739 | 0.6759 |
| 9 | 4 | 0.7188 | 0.7135 | 0.7131 | 0.7219 |
| 9 | 5 | 0.6250 | 0.6224 | 0.6222 | 0.6231 |

## B.4 Experiments on the breast cancer dataset

The full results for the breast cancer dataset can be found in Tables 21, 22, 23, and 24.

**Table 21** Full experimental results for the breast cancer dataset with learning rate $10^{-3}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.7135 | 0.6586 | 0.6652 | 0.7693 |
| 2 | 1 | 0.1927 | 0.1829 | 0.1795 | 0.1875 |
| 2 | 2 | 0.3490 | 0.3146 | 0.3973 | 0.3396 |
| 3 | 1 | 0.4115 | 0.3977 | 0.3973 | 0.3983 |
| 3 | 2 | 0.1719 | 0.1713 | 0.1723 | 0.1772 |
| 3 | 3 | 0.6615 | 0.5882 | 0.6080 | 0.6939 |
| 4 | 1 | 0.9115 | 0.9094 | 0.9116 | 0.9076 |
| 4 | 2 | 0.4219 | 0.3041 | 0.5045 | 0.7094 |
| 4 | 3 | 0.0781 | 0.0781 | 0.0813 | 0.0797 |
| 4 | 4 | 0.3958 | 0.3956 | 0.4036 | 0.4048 |
| 5 | 1 | 0.0781 | 0.0779 | 0.0830 | 0.0781 |
| 5 | 2 | 0.1510 | 0.1364 | 0.1777 | 0.1167 |
| 5 | 3 | 0.5729 | 0.3642 | 0.4911 | 0.2895 |
| 5 | 4 | 0.3958 | 0.3845 | 0.4304 | 0.4163 |
| 5 | 5 | 0.7031 | 0.7018 | 0.7116 | 0.7059 |
| 6 | 1 | 0.4063 | 0.3968 | 0.3964 | 0.3986 |
| 6 | 2 | 0.9167 | 0.9120 | 0.9036 | 0.9288 |
| 6 | 3 | 0.3906 | 0.3886 | 0.4116 | 0.4067 |
| 6 | 4 | 0.4375 | 0.4063 | 0.4107 | 0.4051 |
| 6 | 5 | 0.4219 | 0.3742 | 0.3866 | 0.3705 |
| 6 | 6 | 0.5938 | 0.5667 | 0.5679 | 0.5742 |
| 7 | 1 | 0.4844 | 0.4706 | 0.4705 | 0.4706 |
| 7 | 2 | 0.5677 | 0.3621 | 0.4866 | 0.2884 |
| 7 | 3 | 0.3021 | 0.3018 | 0.3071 | 0.3095 |
| 7 | 4 | 0.5104 | 0.4905 | 0.4911 | 0.4908 |
| 7 | 5 | 0.3229 | 0.3203 | 0.3429 | 0.3330 |
| 7 | 6 | 0.6771 | 0.6679 | 0.6679 | 0.6679 |
| 7 | 7 | 0.4948 | 0.4421 | 0.4563 | 0.4468 |
| 8 | 1 | 0.1615 | 0.1587 | 0.1759 | 0.1578 |
| 8 | 2 | 0.4115 | 0.3120 | 0.4884 | 0.4197 |
| 8 | 3 | 0.5729 | 0.5729 | 0.5911 | 0.5918 |
| 8 | 4 | 0.4531 | 0.4530 | 0.4688 | 0.4684 |
| 8 | 5 | 0.7500 | 0.7209 | 0.7161 | 0.7727 |
| 8 | 6 | 0.6927 | 0.6607 | 0.6598 | 0.6944 |
| 8 | 7 | 0.5938 | 0.4976 | 0.5357 | 0.5697 |
| 9 | 1 | 0.2500 | 0.2493 | 0.2625 | 0.2563 |
| 9 | 2 | 0.1979 | 0.1971 | 0.1982 | 0.2034 |
| 9 | 3 | 0.2917 | 0.2401 | 0.2554 | 0.2280 |
| 9 | 4 | 0.6823 | 0.6135 | 0.6295 | 0.7324 |
| 9 | 5 | 0.5156 | 0.4651 | 0.4777 | 0.4729 |
| 9 | 6 | 0.6094 | 0.5514 | 0.5652 | 0.5912 |
| 9 | 7 | 0.5313 | 0.5211 | 0.5214 | 0.5212 |
| 10 | 1 | 0.0573 | 0.0571 | 0.0616 | 0.0564 |
| 10 | 2 | 0.5573 | 0.4056 | 0.4866 | 0.4548 |
| 10 | 3 | 0.2292 | 0.2292 | 0.2357 | 0.2357 |
| 10 | 4 | 0.6094 | 0.4599 | 0.5366 | 0.6647 |
| 10 | 5 | 0.3698 | 0.3332 | 0.3402 | 0.3290 |
| 10 | 6 | 0.5729 | 0.3857 | 0.4946 | 0.4570 |
| 10 | 7 | 0.5521 | 0.5104 | 0.5179 | 0.5210 |
| 11 | 1 | 0.1615 | 0.1513 | 0.1848 | 0.1401 |
| 11 | 2 | 0.2500 | 0.2471 | 0.2679 | 0.2533 |
| 11 | 3 | 0.5729 | 0.4050 | 0.4982 | 0.4912 |
| 11 | 4 | 0.3802 | 0.3764 | 0.4045 | 0.3963 |
| 11 | 5 | 0.4323 | 0.4322 | 0.4420 | 0.4425 |
| 11 | 6 | 0.5677 | 0.4992 | 0.5205 | 0.5297 |
| 11 | 7 | 0.6823 | 0.6440 | 0.6455 | 0.6861 |

**Table 22** Full experimental results for the breast cancer dataset with learning rate $10^{-2}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5885 | 0.4552 | 0.5205 | 0.5618 |
| 2 | 1 | 0.4427 | 0.3485 | 0.3902 | 0.3351 |
| 2 | 2 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 3 | 1 | 0.0625 | 0.0621 | 0.0607 | 0.0661 |
| 3 | 2 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 3 | 3 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 4 | 1 | 0.5260 | 0.3544 | 0.4527 | 0.3178 |
| 4 | 2 | 0.2396 | 0.2366 | 0.2357 | 0.2426 |
| 4 | 3 | 0.5573 | 0.3579 | 0.4777 | 0.2861 |
| 4 | 4 | 0.6042 | 0.4210 | 0.5250 | 0.7979 |
| 5 | 1 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 5 | 2 | 0.2083 | 0.1827 | 0.1839 | 0.1815 |
| 5 | 3 | 0.5938 | 0.4425 | 0.5214 | 0.5889 |
| 5 | 4 | 0.5781 | 0.4335 | 0.5080 | 0.5271 |
| 5 | 5 | 0.6198 | 0.4743 | 0.5473 | 0.7130 |
| 6 | 1 | 0.0990 | 0.0983 | 0.0973 | 0.1033 |
| 6 | 2 | 0.6198 | 0.4743 | 0.5473 | 0.7130 |
| 6 | 3 | 0.6198 | 0.4662 | 0.5455 | 0.7477 |
| 6 | 4 | 0.5052 | 0.4104 | 0.4509 | 0.4199 |
| 6 | 5 | 0.5885 | 0.3820 | 0.5063 | 0.7932 |
| 6 | 6 | 0.5729 | 0.3642 | 0.4911 | 0.2895 |
| 7 | 1 | 0.1146 | 0.1084 | 0.1036 | 0.1146 |
| 7 | 2 | 0.7500 | 0.7064 | 0.7054 | 0.8144 |
| 7 | 3 | 0.8750 | 0.8634 | 0.8500 | 0.9118 |
| 7 | 4 | 0.5208 | 0.4690 | 0.4821 | 0.4780 |

**Table 22** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 7 | 5 | 0.4323 | 0.3094 | 0.3723 | 0.2708 |
| 7 | 6 | 0.8333 | 0.8212 | 0.8125 | 0.8474 |
| 7 | 7 | 0.6875 | 0.5944 | 0.6250 | 0.8256 |
| 8 | 1 | 0.0677 | 0.0677 | 0.0688 | 0.0702 |
| 8 | 2 | 0.6302 | 0.4808 | 0.5563 | 0.8060 |
| 8 | 3 | 0.8906 | 0.8816 | 0.8688 | 0.9211 |
| 8 | 4 | 0.5990 | 0.4369 | 0.5241 | 0.6311 |
| 8 | 5 | 0.6198 | 0.4662 | 0.5455 | 0.7477 |
| 8 | 6 | 0.6406 | 0.5175 | 0.5723 | 0.7441 |
| 8 | 7 | 0.6198 | 0.4743 | 0.5473 | 0.7130 |
| 9 | 1 | 0.5521 | 0.3557 | 0.4732 | 0.2849 |
| 9 | 2 | 0.5885 | 0.3820 | 0.5063 | 0.7932 |
| 9 | 3 | 0.5990 | 0.4082 | 0.5188 | 0.7963 |
| 9 | 4 | 0.5781 | 0.3663 | 0.4955 | 0.2906 |
| 9 | 5 | 0.5990 | 0.4082 | 0.5188 | 0.7963 |
| 9 | 6 | 0.5885 | 0.3820 | 0.5063 | 0.7932 |
| 9 | 7 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 10 | 1 | 0.5573 | 0.3579 | 0.4777 | 0.2861 |
| 10 | 2 | 0.5625 | 0.3600 | 0.4821 | 0.2872 |
| 10 | 3 | 0.6979 | 0.6856 | 0.6839 | 0.6887 |
| 10 | 4 | 0.6302 | 0.5169 | 0.5652 | 0.6777 |
| 10 | 5 | 0.5885 | 0.4032 | 0.5098 | 0.5941 |
| 10 | 6 | 0.6563 | 0.5351 | 0.5875 | 0.8146 |
| 10 | 7 | 0.7969 | 0.7668 | 0.7580 | 0.8609 |
| 11 | 1 | 0.1250 | 0.1172 | 0.1125 | 0.1231 |
| 11 | 2 | 0.5469 | 0.3638 | 0.4705 | 0.3397 |
| 11 | 3 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 11 | 4 | 0.8125 | 0.7856 | 0.7750 | 0.8784 |
| 11 | 5 | 0.6927 | 0.6556 | 0.6563 | 0.6998 |
| 11 | 6 | 0.6354 | 0.4921 | 0.5625 | 0.8077 |
| 11 | 7 | 0.5885 | 0.3820 | 0.5063 | 0.7932 |

**Table 23** Full experimental results for the breast cancer dataset with learning rate $10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 2 | 1 | 0.9531 | 0.9513 | 0.9473 | 0.9568 |
| 2 | 2 | 0.5833 | 0.4365 | 0.5125 | 0.5449 |
| 3 | 1 | 0.9531 | 0.9515 | 0.9491 | 0.9544 |
| 3 | 2 | 0.4427 | 0.3485 | 0.5205 | 0.6421 |
| 3 | 3 | 0.7396 | 0.7190 | 0.7143 | 0.7424 |
| 4 | 1 | 0.8594 | 0.8496 | 0.8402 | 0.8750 |
| 4 | 2 | 0.8854 | 0.8784 | 0.8696 | 0.8983 |
| 4 | 3 | 0.8229 | 0.8100 | 0.8018 | 0.8355 |
| 4 | 4 | 0.8281 | 0.8199 | 0.8152 | 0.8287 |

**Table 23** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 5 | 1 | 0.9167 | 0.9133 | 0.9089 | 0.9196 |
| 5 | 2 | 0.9167 | 0.9143 | 0.9143 | 0.9143 |
| 5 | 3 | 0.8698 | 0.8607 | 0.8509 | 0.8869 |
| 5 | 4 | 0.8854 | 0.8802 | 0.8750 | 0.8889 |
| 5 | 5 | 0.9271 | 0.9238 | 0.9179 | 0.9333 |
| 6 | 1 | 0.6146 | 0.6104 | 0.6143 | 0.6113 |
| 6 | 2 | 0.4063 | 0.3764 | 0.3804 | 0.3745 |
| 6 | 3 | 0.5729 | 0.5682 | 0.6071 | 0.6212 |
| 6 | 4 | 0.9167 | 0.9151 | 0.9196 | 0.9125 |
| 6 | 5 | 0.9063 | 0.9005 | 0.8911 | 0.9214 |
| 6 | 6 | 0.7708 | 0.7699 | 0.7821 | 0.7748 |
| 7 | 1 | 0.9323 | 0.9300 | 0.9277 | 0.9327 |
| 7 | 2 | 0.9271 | 0.9247 | 0.9232 | 0.9265 |
| 7 | 3 | 0.8229 | 0.8165 | 0.8143 | 0.8194 |
| 7 | 4 | 0.9063 | 0.9020 | 0.8964 | 0.9111 |
| 7 | 5 | 0.8594 | 0.8577 | 0.8652 | 0.8560 |
| 7 | 6 | 0.8958 | 0.8929 | 0.8929 | 0.8929 |
| 7 | 7 | 0.9063 | 0.9011 | 0.8929 | 0.9175 |
| 8 | 1 | 0.9479 | 0.9458 | 0.9411 | 0.9526 |
| 8 | 2 | 0.8125 | 0.8064 | 0.8054 | 0.8077 |
| 8 | 3 | 0.9375 | 0.9355 | 0.9339 | 0.9372 |
| 8 | 4 | 0.8906 | 0.8864 | 0.8830 | 0.8911 |
| 8 | 5 | 0.9479 | 0.9462 | 0.9446 | 0.9480 |
| 8 | 6 | 0.9063 | 0.9015 | 0.8946 | 0.9141 |
| 8 | 7 | 0.9063 | 0.9028 | 0.9000 | 0.9065 |
| 9 | 1 | 0.9323 | 0.9309 | 0.9348 | 0.9284 |
| 9 | 2 | 0.8854 | 0.8812 | 0.8786 | 0.8848 |
| 9 | 3 | 0.9115 | 0.9091 | 0.9098 | 0.9084 |
| 9 | 4 | 0.9167 | 0.9140 | 0.9125 | 0.9157 |
| 9 | 5 | 0.9271 | 0.9238 | 0.9179 | 0.9333 |
| 9 | 6 | 0.8958 | 0.8906 | 0.8839 | 0.9028 |
| 9 | 7 | 0.8802 | 0.8726 | 0.8634 | 0.8944 |
| 10 | 1 | 0.9010 | 0.8996 | 0.9063 | 0.8971 |
| 10 | 2 | 0.9271 | 0.9238 | 0.9179 | 0.9333 |
| 10 | 3 | 0.8802 | 0.8751 | 0.8705 | 0.8822 |
| 10 | 4 | 0.9167 | 0.9125 | 0.9054 | 0.9253 |
| 10 | 5 | 0.8698 | 0.8591 | 0.8473 | 0.8966 |
| 10 | 6 | 0.9531 | 0.9513 | 0.9473 | 0.9568 |
| 10 | 7 | 0.9010 | 0.8972 | 0.8938 | 0.9021 |
| 11 | 1 | 0.7708 | 0.7581 | 0.7536 | 0.7694 |
| 11 | 2 | 0.9167 | 0.9120 | 0.9036 | 0.9288 |
| 11 | 3 | 0.9323 | 0.9291 | 0.9223 | 0.9405 |
| 11 | 4 | 0.9063 | 0.9005 | 0.8911 | 0.9214 |
| 11 | 5 | 0.8802 | 0.8756 | 0.8723 | 0.8802 |
| 11 | 6 | 0.8906 | 0.8854 | 0.8795 | 0.8958 |
| 11 | 7 | 0.9219 | 0.9198 | 0.9205 | 0.9191 |

**Table 24** Full experimental results for the breast cancer dataset with learning rate $2 * 10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5833 | 0.3684 | 0.5000 | 0.2917 |
| 2 | 1 | 0.5156 | 0.5150 | 0.5241 | 0.5236 |
| 2 | 2 | 0.5938 | 0.4057 | 0.5143 | 0.6702 |
| 3 | 1 | 0.9427 | 0.9402 | 0.9348 | 0.9485 |
| 3 | 2 | 0.8125 | 0.8104 | 0.8179 | 0.8096 |
| 3 | 3 | 0.8854 | 0.8817 | 0.8804 | 0.8833 |
| 4 | 1 | 0.7969 | 0.7881 | 0.7848 | 0.7938 |
| 4 | 2 | 0.5417 | 0.5186 | 0.5196 | 0.5206 |
| 4 | 3 | 0.9167 | 0.9143 | 0.9143 | 0.9143 |
| 4 | 4 | 0.9688 | 0.9746 | 0.9675 | 0.9625 |
| 5 | 1 | 0.9323 | 0.9297 | 0.9259 | 0.9349 |
| 5 | 2 | 0.4115 | 0.4095 | 0.4330 | 0.4293 |
| 5 | 3 | 0.9219 | 0.9198 | 0.9205 | 0.9191 |
| 5 | 4 | 0.9375 | 0.9352 | 0.9321 | 0.9392 |
| 5 | 5 | 0.8125 | 0.7951 | 0.7857 | 0.8361 |
| 6 | 1 | 0.9375 | 0.9355 | 0.9339 | 0.9372 |
| 6 | 2 | 0.7240 | 0.6993 | 0.6955 | 0.7277 |
| 6 | 3 | 0.8698 | 0.8615 | 0.8527 | 0.8828 |
| 6 | 4 | 0.9167 | 0.9153 | 0.9214 | 0.9126 |
| 6 | 5 | 0.9219 | 0.9185 | 0.9134 | 0.9264 |
| 6 | 6 | 0.9115 | 0.9084 | 0.9063 | 0.9111 |
| 7 | 1 | 0.9531 | 0.9513 | 0.9473 | 0.9568 |
| 7 | 2 | 0.9271 | 0.9244 | 0.9214 | 0.9283 |
| 7 | 3 | 0.8177 | 0.8128 | 0.8134 | 0.8123 |
| 7 | 4 | 0.9635 | 0.9621 | 0.9580 | 0.9677 |
| 7 | 5 | 0.9219 | 0.9189 | 0.9152 | 0.9239 |
| 7 | 6 | 0.9219 | 0.9181 | 0.9116 | 0.9293 |
| 7 | 7 | 0.8958 | 0.8936 | 0.8964 | 0.8915 |
| 8 | 1 | 0.9323 | 0.9291 | 0.9223 | 0.9405 |
| 8 | 2 | 0.9167 | 0.9140 | 0.9125 | 0.9157 |
| 8 | 3 | 0.9167 | 0.9125 | 0.9054 | 0.9253 |
| 8 | 4 | 0.9427 | 0.9410 | 0.9402 | 0.9418 |
| 8 | 5 | 0.9375 | 0.9344 | 0.9268 | 0.9478 |
| 8 | 6 | 0.9375 | 0.9355 | 0.9339 | 0.9372 |
| 8 | 7 | 0.9115 | 0.9080 | 0.9045 | 0.9130 |
| 9 | 1 | 0.9427 | 0.9407 | 0.9384 | 0.9436 |
| 9 | 2 | 0.9479 | 0.9458 | 0.9411 | 0.9526 |
| 9 | 3 | 0.9167 | 0.9133 | 0.9089 | 0.9196 |
| 9 | 4 | 0.9323 | 0.9302 | 0.9295 | 0.9311 |
| 9 | 5 | 0.9271 | 0.9255 | 0.9286 | 0.9233 |
| 9 | 6 | 0.9427 | 0.9412 | 0.9420 | 0.9404 |
| 9 | 7 | 0.9167 | 0.9125 | 0.9054 | 0.9253 |
| 10 | 1 | 0.9427 | 0.9402 | 0.9348 | 0.9485 |
| 10 | 2 | 0.9271 | 0.9250 | 0.9250 | 0.9250 |
| 10 | 3 | 0.9063 | 0.9036 | 0.9036 | 0.9036 |
| 10 | 4 | 0.9375 | 0.9350 | 0.9304 | 0.9416 |
| 10 | 5 | 0.9219 | 0.9192 | 0.9170 | 0.9219 |
| 10 | 6 | 0.9479 | 0.9462 | 0.9446 | 0.9480 |
| 10 | 7 | 0.9010 | 0.8958 | 0.8884 | 0.9100 |
| 11 | 1 | 0.9427 | 0.9405 | 0.9366 | 0.9458 |
| 11 | 2 | 0.9115 | 0.9076 | 0.9027 | 0.9153 |
| 11 | 3 | 0.8906 | 0.8843 | 0.8759 | 0.9021 |
| 11 | 4 | 0.9531 | 0.9515 | 0.9491 | 0.9544 |
| 11 | 5 | 0.9271 | 0.9244 | 0.9214 | 0.9283 |
| 11 | 6 | 0.9323 | 0.9305 | 0.9313 | 0.9298 |
| 11 | 7 | 0.9323 | 0.9297 | 0.9259 | 0.9349 |

## B.5 Experiments on the adult dataset

The full results for the adult dataset can be found in Tables 25, 26, 27, and 28.

**Table 25** Full experimental results for the adult dataset with learning rate $10^{-3}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.6471 | 0.6329 | 0.6471 | 0.6740 |
| 2 | 1 | 0.3854 | 0.3811 | 0.3854 | 0.3821 |
| 2 | 2 | 0.5599 | 0.5077 | 0.5599 | 0.6040 |
| 3 | 1 | 0.4258 | 0.3966 | 0.4258 | 0.4080 |
| 3 | 2 | 0.3828 | 0.3717 | 0.3828 | 0.3739 |
| 3 | 3 | 0.6367 | 0.6336 | 0.6367 | 0.6416 |
| 4 | 1 | 0.4570 | 0.4301 | 0.4570 | 0.4470 |
| 4 | 2 | 0.5130 | 0.4281 | 0.5130 | 0.5321 |
| 4 | 3 | 0.4635 | 0.3328 | 0.4635 | 0.3312 |
| 4 | 4 | 0.4141 | 0.4130 | 0.4141 | 0.4135 |
| 5 | 1 | 0.4596 | 0.4596 | 0.4596 | 0.4596 |
| 5 | 2 | 0.5143 | 0.3644 | 0.5143 | 0.7536 |
| 5 | 3 | 0.3190 | 0.3005 | 0.3190 | 0.2976 |
| 5 | 4 | 0.4635 | 0.4324 | 0.4635 | 0.4533 |
| 5 | 5 | 0.4453 | 0.4083 | 0.4453 | 0.4271 |
| 6 | 1 | 0.2682 | 0.2682 | 0.2682 | 0.2682 |
| 6 | 2 | 0.4857 | 0.3335 | 0.4857 | 0.3346 |
| 6 | 3 | 0.3424 | 0.3412 | 0.3424 | 0.3413 |
| 6 | 4 | 0.5208 | 0.5115 | 0.5208 | 0.5226 |
| 6 | 5 | 0.5885 | 0.5611 | 0.5885 | 0.6181 |
| 6 | 6 | 0.5964 | 0.5844 | 0.5964 | 0.6088 |
| 7 | 1 | 0.4492 | 0.4153 | 0.4492 | 0.4339 |
| 7 | 2 | 0.4922 | 0.4895 | 0.4922 | 0.4920 |
| 7 | 3 | 0.2760 | 0.2760 | 0.2760 | 0.2760 |
| 7 | 4 | 0.3451 | 0.3451 | 0.3451 | 0.3451 |

**Table 25** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 7 | 5 | 0.4388 | 0.4359 | 0.4388 | 0.4375 |
| 7 | 6 | 0.5339 | 0.5290 | 0.5339 | 0.5353 |
| 7 | 7 | 0.5924 | 0.5893 | 0.5924 | 0.5954 |
| 8 | 1 | 0.2474 | 0.2414 | 0.2474 | 0.2392 |
| 8 | 2 | 0.4089 | 0.3834 | 0.4089 | 0.3908 |
| 8 | 3 | 0.4375 | 0.4371 | 0.4375 | 0.4373 |
| 8 | 4 | 0.6549 | 0.6538 | 0.6549 | 0.6571 |
| 8 | 5 | 0.4753 | 0.4736 | 0.4753 | 0.4749 |
| 8 | 6 | 0.5690 | 0.5657 | 0.5690 | 0.5712 |
| 8 | 7 | 0.4987 | 0.4817 | 0.4987 | 0.4985 |
| 9 | 1 | 0.3477 | 0.3371 | 0.3477 | 0.3373 |
| 9 | 2 | 0.4935 | 0.4030 | 0.4935 | 0.4835 |
| 9 | 3 | 0.3281 | 0.2952 | 0.3281 | 0.2886 |
| 9 | 4 | 0.3841 | 0.3832 | 0.3841 | 0.3834 |
| 9 | 5 | 0.3984 | 0.3839 | 0.3984 | 0.3878 |
| 9 | 6 | 0.5117 | 0.5115 | 0.5117 | 0.5117 |
| 9 | 7 | 0.6797 | 0.6774 | 0.6797 | 0.6848 |
| 10 | 1 | 0.4909 | 0.4843 | 0.4909 | 0.4904 |
| 10 | 2 | 0.5013 | 0.3362 | 0.5013 | 0.7503 |
| 10 | 3 | 0.4844 | 0.3587 | 0.4844 | 0.4277 |
| 10 | 4 | 0.5443 | 0.4721 | 0.5443 | 0.5977 |
| 10 | 5 | 0.5208 | 0.4991 | 0.5208 | 0.5252 |
| 10 | 6 | 0.4844 | 0.4222 | 0.4844 | 0.4726 |
| 10 | 7 | 0.4701 | 0.4679 | 0.4701 | 0.4696 |
| 11 | 1 | 0.4010 | 0.3966 | 0.4010 | 0.3980 |
| 11 | 2 | 0.3346 | 0.3008 | 0.3346 | 0.2949 |
| 11 | 3 | 0.5000 | 0.3333 | 0.5000 | 0.2500 |
| 11 | 4 | 0.6094 | 0.5926 | 0.6094 | 0.6310 |
| 11 | 5 | 0.5026 | 0.4313 | 0.5026 | 0.5052 |
| 11 | 6 | 0.4909 | 0.4148 | 0.4909 | 0.4810 |
| 11 | 7 | 0.6146 | 0.6135 | 0.6146 | 0.6158 |

**Table 26** Full experimental results for the adult dataset with learning rate $10^{-2}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.4219 | 0.3722 | 0.4219 | 0.3857 |
| 2 | 1 | 0.5990 | 0.5978 | 0.5990 | 0.6002 |
| 2 | 2 | 0.3060 | 0.2871 | 0.3060 | 0.2830 |
| 3 | 1 | 0.3008 | 0.2805 | 0.3008 | 0.2754 |
| 3 | 2 | 0.4154 | 0.3905 | 0.4154 | 0.3989 |
| 3 | 3 | 0.4909 | 0.3782 | 0.4909 | 0.4668 |
| 4 | 1 | 0.5091 | 0.4565 | 0.5091 | 0.5149 |
| 4 | 2 | 0.5078 | 0.4273 | 0.5078 | 0.5179 |
| 4 | 3 | 0.5794 | 0.5639 | 0.5794 | 0.5926 |
| 4 | 4 | 0.6432 | 0.6395 | 0.6432 | 0.6494 |

**Table 26** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 5 | 1 | 0.3372 | 0.3180 | 0.3372 | 0.3165 |
| 5 | 2 | 0.2904 | 0.2898 | 0.2904 | 0.2897 |
| 5 | 3 | 0.4961 | 0.4935 | 0.4961 | 0.4960 |
| 5 | 4 | 0.7188 | 0.7157 | 0.7188 | 0.7287 |
| 5 | 5 | 0.5911 | 0.5893 | 0.5911 | 0.5928 |
| 6 | 1 | 0.4531 | 0.3928 | 0.4531 | 0.4222 |
| 6 | 2 | 0.7005 | 0.6999 | 0.7005 | 0.7021 |
| 6 | 3 | 0.6107 | 0.6008 | 0.6107 | 0.6229 |
| 6 | 4 | 0.5924 | 0.5858 | 0.5924 | 0.5987 |
| 6 | 5 | 0.6289 | 0.6289 | 0.6289 | 0.6289 |
| 6 | 6 | 0.6901 | 0.6852 | 0.6901 | 0.7028 |
| 7 | 1 | 0.4258 | 0.3619 | 0.4258 | 0.3762 |
| 7 | 2 | 0.3190 | 0.3106 | 0.3190 | 0.3097 |
| 7 | 3 | 0.7083 | 0.7060 | 0.7083 | 0.7151 |
| 7 | 4 | 0.6836 | 0.6834 | 0.6836 | 0.6840 |
| 7 | 5 | 0.6107 | 0.6106 | 0.6107 | 0.6108 |
| 7 | 6 | 0.5729 | 0.5728 | 0.5729 | 0.5730 |
| 7 | 7 | 0.5951 | 0.5933 | 0.5951 | 0.5968 |
| 8 | 1 | 0.4987 | 0.3328 | 0.4987 | 0.2497 |
| 8 | 2 | 0.5872 | 0.5630 | 0.5872 | 0.6122 |
| 8 | 3 | 0.5182 | 0.5158 | 0.5182 | 0.5186 |
| 8 | 4 | 0.6289 | 0.6231 | 0.6289 | 0.6373 |
| 8 | 5 | 0.5924 | 0.5899 | 0.5924 | 0.5948 |
| 8 | 6 | 0.6497 | 0.6497 | 0.6497 | 0.6499 |
| 8 | 7 | 0.6797 | 0.6797 | 0.6797 | 0.6797 |
| 9 | 1 | 0.6615 | 0.6603 | 0.6615 | 0.6636 |
| 9 | 2 | 0.5247 | 0.3919 | 0.5247 | 0.6964 |
| 9 | 3 | 0.6146 | 0.5593 | 0.6146 | 0.7300 |
| 9 | 4 | 0.6419 | 0.6413 | 0.6419 | 0.6429 |
| 9 | 5 | 0.5781 | 0.5679 | 0.5781 | 0.5863 |
| 9 | 6 | 0.6745 | 0.6738 | 0.6745 | 0.6759 |
| 9 | 7 | 0.5326 | 0.5214 | 0.5326 | 0.5359 |
| 10 | 1 | 0.5104 | 0.4622 | 0.5104 | 0.5162 |
| 10 | 2 | 0.5742 | 0.5571 | 0.5742 | 0.5878 |
| 10 | 3 | 0.5013 | 0.4094 | 0.5013 | 0.5035 |
| 10 | 4 | 0.6211 | 0.6063 | 0.6211 | 0.6426 |
| 10 | 5 | 0.6367 | 0.6366 | 0.6367 | 0.6368 |
| 10 | 6 | 0.7305 | 0.7304 | 0.7305 | 0.7308 |
| 10 | 7 | 0.6849 | 0.6830 | 0.6849 | 0.6895 |
| 11 | 1 | 0.5339 | 0.4624 | 0.5339 | 0.5723 |
| 11 | 2 | 0.7409 | 0.7393 | 0.7409 | 0.7467 |
| 11 | 3 | 0.4896 | 0.4440 | 0.4896 | 0.4845 |
| 11 | 4 | 0.4688 | 0.4516 | 0.4688 | 0.4643 |
| 11 | 5 | 0.5833 | 0.5824 | 0.5833 | 0.5841 |
| 11 | 6 | 0.5521 | 0.5515 | 0.5521 | 0.5524 |
| 11 | 7 | 0.6628 | 0.6592 | 0.6628 | 0.6700 |

**Table 27** Full experimental results for the adult dataset with learning rate $10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.6120 | 0.5867 | 0.6120 | 0.6483 |
| 2 | 1 | 0.5404 | 0.4401 | 0.5404 | 0.6423 |
| 2 | 2 | 0.4961 | 0.4698 | 0.4961 | 0.4951 |
| 3 | 1 | 0.5143 | 0.3879 | 0.5143 | 0.5823 |
| 3 | 2 | 0.5729 | 0.5384 | 0.5729 | 0.6040 |
| 3 | 3 | 0.5247 | 0.5242 | 0.5247 | 0.5248 |
| 4 | 1 | 0.4766 | 0.3249 | 0.4766 | 0.2690 |
| 4 | 2 | 0.7865 | 0.7840 | 0.7865 | 0.8001 |
| 4 | 3 | 0.4909 | 0.4476 | 0.4909 | 0.4867 |
| 4 | 4 | 0.6927 | 0.6848 | 0.6927 | 0.7143 |
| 5 | 1 | 0.7539 | 0.7538 | 0.7539 | 0.7544 |
| 5 | 2 | 0.6940 | 0.6830 | 0.6940 | 0.7252 |
| 5 | 3 | 0.7214 | 0.7213 | 0.7214 | 0.7216 |
| 5 | 4 | 0.5977 | 0.5856 | 0.5977 | 0.6105 |
| 5 | 5 | 0.7435 | 0.7435 | 0.7435 | 0.7436 |
| 6 | 1 | 0.7435 | 0.7386 | 0.7435 | 0.7632 |
| 6 | 2 | 0.6641 | 0.6559 | 0.6641 | 0.6812 |
| 6 | 3 | 0.6615 | 0.6505 | 0.6615 | 0.6846 |
| 6 | 4 | 0.7370 | 0.7359 | 0.7370 | 0.7407 |
| 6 | 5 | 0.6745 | 0.6734 | 0.6745 | 0.6768 |
| 6 | 6 | 0.6549 | 0.6407 | 0.6549 | 0.6842 |
| 7 | 1 | 0.7201 | 0.7134 | 0.7201 | 0.7426 |
| 7 | 2 | 0.5378 | 0.5272 | 0.5378 | 0.5415 |
| 7 | 3 | 0.7435 | 0.7388 | 0.7435 | 0.7624 |
| 7 | 4 | 0.6354 | 0.6281 | 0.6354 | 0.6470 |
| 7 | 5 | 0.7435 | 0.7434 | 0.7435 | 0.7439 |
| 7 | 6 | 0.6250 | 0.6243 | 0.6250 | 0.6259 |
| 7 | 7 | 0.7044 | 0.7039 | 0.7044 | 0.7058 |
| 8 | 1 | 0.7865 | 0.7864 | 0.7865 | 0.7870 |
| 8 | 2 | 0.5352 | 0.5138 | 0.5352 | 0.5427 |
| 8 | 3 | 0.7122 | 0.7052 | 0.7122 | 0.7348 |
| 8 | 4 | 0.7344 | 0.7343 | 0.7344 | 0.7345 |
| 8 | 5 | 0.7122 | 0.7119 | 0.7122 | 0.7131 |
| 8 | 6 | 0.7005 | 0.7002 | 0.7005 | 0.7013 |
| 8 | 7 | 0.6302 | 0.6279 | 0.6302 | 0.6335 |
| 9 | 1 | 0.7409 | 0.7387 | 0.7409 | 0.7494 |
| 9 | 2 | 0.7643 | 0.7643 | 0.7643 | 0.7644 |
| 9 | 3 | 0.7005 | 0.7005 | 0.7005 | 0.7006 |
| 9 | 4 | 0.7135 | 0.7094 | 0.7135 | 0.7265 |
| 9 | 5 | 0.6667 | 0.6661 | 0.6667 | 0.6678 |
| 9 | 6 | 0.7148 | 0.7130 | 0.7148 | 0.7204 |
| 9 | 7 | 0.7266 | 0.7266 | 0.7266 | 0.7266 |
| 10 | 1 | 0.7617 | 0.7616 | 0.7617 | 0.7622 |
| 10 | 2 | 0.7734 | 0.7733 | 0.7734 | 0.7743 |
| 10 | 3 | 0.7266 | 0.7264 | 0.7266 | 0.7272 |
| 10 | 4 | 0.6615 | 0.6567 | 0.6615 | 0.6708 |
| 10 | 5 | 0.6836 | 0.6780 | 0.6836 | 0.6972 |
| 10 | 6 | 0.7135 | 0.7128 | 0.7135 | 0.7157 |
| 10 | 7 | 0.7318 | 0.7306 | 0.7318 | 0.7358 |
| 11 | 1 | 0.5846 | 0.5787 | 0.5846 | 0.5897 |
| 11 | 2 | 0.5938 | 0.5905 | 0.5938 | 0.5968 |
| 11 | 3 | 0.6888 | 0.6866 | 0.6888 | 0.6944 |
| 11 | 4 | 0.6445 | 0.6442 | 0.6445 | 0.6451 |
| 11 | 5 | 0.7083 | 0.7067 | 0.7083 | 0.7132 |
| 11 | 6 | 0.7174 | 0.7174 | 0.7174 | 0.7175 |
| 11 | 7 | 0.6836 | 0.6834 | 0.6836 | 0.6840 |

**Table 28** Full experimental results for the adult dataset with learning rate $2 * 10^{-1}$

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 1 | 1 | 0.5000 | 0.3333 | 0.5000 | 0.2500 |
| 2 | 1 | 0.5378 | 0.4854 | 0.5378 | 0.5637 |
| 2 | 2 | 0.5169 | 0.4964 | 0.5169 | 0.5202 |
| 3 | 1 | 0.4570 | 0.3758 | 0.4570 | 0.4104 |
| 3 | 2 | 0.5169 | 0.3857 | 0.5169 | 0.6165 |
| 3 | 3 | 0.6276 | 0.6274 | 0.6276 | 0.6278 |
| 4 | 1 | 0.6862 | 0.6843 | 0.6862 | 0.6907 |
| 4 | 2 | 0.6081 | 0.6041 | 0.6081 | 0.6126 |
| 4 | 3 | 0.6237 | 0.6077 | 0.6237 | 0.6478 |
| 4 | 4 | 0.6432 | 0.6406 | 0.6432 | 0.6476 |
| 5 | 1 | 0.7201 | 0.7199 | 0.7201 | 0.7206 |
| 5 | 2 | 0.5768 | 0.5678 | 0.5768 | 0.5838 |
| 5 | 3 | 0.6693 | 0.6658 | 0.6693 | 0.6766 |
| 5 | 4 | 0.6849 | 0.6830 | 0.6849 | 0.6895 |
| 5 | 5 | 0.6810 | 0.6810 | 0.6810 | 0.6810 |
| 6 | 1 | 0.6732 | 0.6693 | 0.6732 | 0.6817 |
| 6 | 2 | 0.6445 | 0.6443 | 0.6445 | 0.6450 |
| 6 | 3 | 0.7331 | 0.7319 | 0.7331 | 0.7373 |
| 6 | 4 | 0.6953 | 0.6947 | 0.6953 | 0.6969 |
| 6 | 5 | 0.7135 | 0.7132 | 0.7135 | 0.7147 |
| 6 | 6 | 0.6315 | 0.6312 | 0.6315 | 0.6319 |
| 7 | 1 | 0.7201 | 0.7109 | 0.7201 | 0.7521 |
| 7 | 2 | 0.7318 | 0.7315 | 0.7318 | 0.7327 |
| 7 | 3 | 0.6172 | 0.6126 | 0.6172 | 0.6231 |
| 7 | 4 | 0.6758 | 0.6745 | 0.6758 | 0.6787 |
| 7 | 5 | 0.7279 | 0.7252 | 0.7279 | 0.7369 |
| 7 | 6 | 0.5807 | 0.5791 | 0.5807 | 0.5820 |
| 7 | 7 | 0.6745 | 0.6713 | 0.6745 | 0.6816 |
| 8 | 1 | 0.7240 | 0.7227 | 0.7240 | 0.7281 |
| 8 | 2 | 0.6354 | 0.6253 | 0.6354 | 0.6518 |

**Table 28** continued

| Qubits | Layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 8 | 3 | 0.6979 | 0.6979 | 0.6979 | 0.6980 |
| 8 | 4 | 0.5990 | 0.5745 | 0.5990 | 0.6285 |
| 8 | 5 | 0.6719 | 0.6558 | 0.6719 | 0.7114 |
| 8 | 6 | 0.7214 | 0.7213 | 0.7214 | 0.7215 |
| 8 | 7 | 0.5924 | 0.5856 | 0.5924 | 0.5990 |
| 9 | 1 | 0.7109 | 0.7013 | 0.7109 | 0.7422 |
| 9 | 2 | 0.6823 | 0.6735 | 0.6823 | 0.7043 |
| 9 | 3 | 0.7409 | 0.7400 | 0.7409 | 0.7442 |
| 9 | 4 | 0.6875 | 0.6871 | 0.6875 | 0.6885 |
| 9 | 5 | 0.7122 | 0.7106 | 0.7122 | 0.7170 |
| 9 | 6 | 0.6667 | 0.6664 | 0.6667 | 0.6672 |
| 9 | 7 | 0.5872 | 0.5864 | 0.5872 | 0.5880 |
| 10 | 1 | 0.7188 | 0.7008 | 0.7188 | 0.7877 |
| 10 | 2 | 0.7526 | 0.7519 | 0.7526 | 0.7554 |
| 10 | 3 | 0.7214 | 0.7213 | 0.7214 | 0.7215 |
| 10 | 4 | 0.7044 | 0.7044 | 0.7044 | 0.7045 |
| 10 | 5 | 0.6888 | 0.6811 | 0.6888 | 0.7089 |
| 10 | 6 | 0.6484 | 0.6481 | 0.6484 | 0.6489 |
| 10 | 7 | 0.7201 | 0.7200 | 0.7201 | 0.7203 |
| 11 | 1 | 0.7878 | 0.7874 | 0.7878 | 0.7899 |
| 11 | 2 | 0.6198 | 0.6121 | 0.6198 | 0.6301 |
| 11 | 3 | 0.7656 | 0.7656 | 0.7656 | 0.7657 |
| 11 | 4 | 0.7474 | 0.7466 | 0.7474 | 0.7507 |
| 11 | 5 | 0.6849 | 0.6805 | 0.6849 | 0.6956 |
| 11 | 6 | 0.7240 | 0.7234 | 0.7240 | 0.7259 |
| 11 | 7 | 0.6979 | 0.6972 | 0.6979 | 0.6999 |

**Author Contributions** Both authors contributed equally.

**Funding** Open Access funding enabled and organized by Projekt DEAL. Not applicable.

**Data Availability** The code is available on request. The data is publicly available as referenced in the text.

## Declarations

**Conflict of interest** The authors declare no competing interests.



## References

Oliveira N, Praça I, Maia E, Sousa O (2021) Intelligent cyber attack detection and classification for network-based intrusion detection systems. Appl Sci 11(4). https://doi.org/10.3390/app11041674

Azevedo V, Silva C, Dutra I (2022) Quantum transfer learning for breast cancer detection. Quantum Mach Intell 4(1):1–14

Schuld M, Killoran N (2022) Is quantum advantage the right goal for quantum machine learning? PRX Quantum 3:030101. https://doi.org/10.1103/PRXQuantum.3.030101

Chen SYC, Yang CHH, Qi J, Chen PY, Ma X, Goan HS (2020) Variational quantum circuits for deep reinforcement learning. IEEE Access. 8:141007–141024. https://doi.org/10.1109/ACCESS.2020.3010470

Chen SYC, Yang CHH, Qi J, Chen PY, Ma X, Goan HS (2020) Variational quantum circuits for deep reinforcement learning. IEEE Access. 8:141007–141024. https://doi.org/10.1109/ACCESS.2020.3010470

Chen SYC, Huang CM, Hsing CW, Kao YJ (2021) An end-to-end trainable hybrid classical-quantum classifier. Mach Learn Sci Technol 2(4):045021

Dong D, Chen C, Li H, Tarn TJ (2008) Quantum reinforcement learning. IEEE Trans Syst Man Cybern Part B (Cybern) 38(5):1207–1220

Dunjko V, Ge Y, Cirac JI (2018) Computational speedups using small quantum devices. Phys Rev Lett 121:250501. https://doi.org/10.1103/PhysRevLett.121.250501

Meyer N, Ufrecht C, Periyasamy M, Scherer DD, Plinge A, Mutschler C (2022) A survey on quantum reinforcement learning. arXiv arXiv:2211.03464

Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(1):1997–2017

Garcia S, Parmisano A, Erquiaga MJ (2020) IoT-23: a labeled dataset with malicious and benign IoT network traffic. Zenodo. https://doi.org/10.5281/zenodo.4743746

Dunjko V, Ge Y, Cirac JI (2018) Computational speedups using small quantum devices. Phys Rev Lett 121:250501. https://doi.org/10.1103/PhysRevLett.121.250501

Peng T, Harrow AW, Ozols M, Wu X (2020) Simulating large quantum circuits on a small quantum computer. Phys Rev Lett 125:150504. https://doi.org/10.1103/PhysRevLett.125.150504

Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum-enhanced feature spaces. Nat 567(7747):209–212

Hubregtsen T, Pichlmeier J, Stecher P, Bertels K (2021) Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. Quantum Mach Intell 3(1):1–19

Islam M, Chowdhury* M, Khan Z, Khan* SM (2022) Hybrid quantum-classical neural network for cloud-supported in-vehicle cyberattack detection. IEEE Sens Lett 6(4):1–4. https://doi.org/10.1109/LSENS.2022.3153931

Islam M, Chowdhury* M, Khan Z, Khan* SM (2022) Hybrid quantum-classical neural network for cloud-supported in-vehicle cyberattack detection. IEEE Sens Lett 6(4):1–4. https://doi.org/10.1109/LSENS.2022.3153931

Suryotrisongko H, Musashi Y (2022a) Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection. Procedia Comput Sci 197(C):223–229. https://doi.org/10.1016/j.procs.2021.12.135

Suryotrisongko H, Musashi Y () Hybrid quantum deep learning and variational quantum classifier-based model for botnet DGA attack detection. Int J Intell Eng Syst 15(3)
Schuld M, Sweke R, Meyer JJ (2021) Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys Rev A 103:032430. https://doi.org/10.1103/PhysRevA.103.032430
Suryotrisongko H, Musashi Y, Tsuneda A, Sugitani K (2022) Adversarial robustness in hybrid quantum-classical deep learning for botnet dga detection. J Inf Process 30:636–644
Schuld M, Petruccione F (2021) Machine learning with quantum computers. Quantum Sci Technol. Springer
Mitarai K, Negoro M, Kitagawa M, Fujii K (2018) Quantum circuit learning. Phys Rev A 98(3):032309
Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research, vol 80. PMLR, pp 4095–4104. https://proceedings.mlr.press/v80/pham18a.html
Suzuki Y, Yano H, Gao Q, Uno S, Tanaka T, Akiyama M, Yamamoto N (2020) Analysis and synthesis of feature map for kernel-based quantum classifier. Quantum Mach Intell 2(1):1–9
Sim S, Johnson PD, Aspuru-Guzik A (2019) Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Adv Quant Technol 2(12):1900070
Peng T, Harrow AW, Ozols M, Wu X (2020) Simulating large quantum circuits on a small quantum computer. Phys Rev Lett 125:150504. https://doi.org/10.1103/PhysRevLett.125.150504
Mitarai K, Negoro M, Kitagawa M, Fujii K (2018) Quantum circuit learning. Phys Rev A 98(3):032309
Rebentrost P, Mohseni M, Lloyd S (2014) Quantum support vector machine for big data classification. Phys Rev Lett 113(13):130503
Schuld M, Killoran N (2019) Quantum machine learning in feature Hilbert spaces. Phys Rev Lett 122:040504. https://doi.org/10.1103/PhysRevLett.122.040504
Schade R, Bauer C, Tamoev K, Mazur L, Plessl C, Kühne TD (2022) Parallel quantum chemistry on noisy intermediate-scale quantum computers. Phys Rev Res 4:033160. https://doi.org/10.1103/PhysRevResearch.4.033160
Schuld M, Killoran N (2019) Quantum machine learning in feature Hilbert spaces. Phys Rev Lett 122:040504. https://doi.org/10.1103/PhysRevLett.122.040504
Schuld M, Killoran N (2022) Is quantum advantage the right goal for quantum machine learning? PRX Quantum 3:030101. https://doi.org/10.1103/PRXQuantum.3.030101
Schuld M, Bocharov A, Svore KM, Wiebe N (2020) Circuit-centric quantum classifiers. Phys Rev A 101:032308. https://doi.org/10.1103/PhysRevA.101.032308
Schuld M, Sweke R, Meyer JJ (2021) Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys Rev A 103:032430. https://doi.org/10.1103/PhysRevA.103.032430
Resch S, Gutierrez A, Huh JS, Bharadwaj S, Eckert Y, Loh G, Oskin M, Tannu S (2021) Accelerating variational quantum algorithms using circuit concurrency. arXiv preprint arXiv:2109.01714
Sim S, Johnson PD, Aspuru-Guzik A (2019) Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Adv Quant Technol 2(12):1900070
Suryotrisongko H, Musashi Y, Tsuneda A, Sugitani K (2022) Adversarial robustness in hybrid quantum-classical deep learning for botnet DGA detection. J Inf Process 30:636–644
Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd international conference for learning representations, San Diego, 2015. http://arxiv.org/abs/1412.6980
Wu SL, Chan J, Guan W, Sun S, Wang A, Zhou C, Livny M, Carminati F, Di Meglio A, Li AC et al (2021) Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. J Phys G Nuclear Part Phys 48(12):125003
Suzuki Y, Yano H, Gao Q, Uno S, Tanaka T, Akiyama M, Yamamoto N (2020) Analysis and synthesis of feature map for kernel-based quantum classifier. Quantum Mach Intell 2(1):1–9
Wang L, Jones R (2021) Big data analytics in cyber security: network traffic and attacks. J Comput Inf Syst 61(5):410–417. https://doi.org/10.1080/08874417.2019.1688731
Janosi A, Steinbrunn W, Pfisterer M, Detrano R (1989) Heart disease. UCI machine learning repository. https://doi.org/10.24432/C52P4X
Wolberg W, Mangasarian O, Street N, Street W (1993) Breast cancer Wisconsin (diagnostic). UCI machine learning repository. https://doi.org/10.24432/C5DW2B
Wu SL, Chan J, Guan W, Sun S, Wang A, Zhou C, Livny M, Carminati F, Di Meglio A, Li AC et al (2021) Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. J Phys G Nuclear Part Phys 48(12):125003