IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 December 2022, accepted 26 December 2022, date of publication 12 January 2023, date of current version 20 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236409

APPLIED RESEARCH

Experimental Evaluation of Quantum Machine Learning Algorithms

RICARDO DANIEL MONTEIRO SIMÕES[1], PATRICK HUBER[1], NICOLA MEIER[1], NIKITA SMAILOV[1], RUDOLF M. FÜCHSLIN[1,2], AND KURT STOCKINGER[1]

[1]School of Engineering, ZHAW Zurich University of Applied Sciences, 8401 Winterthur, Switzerland
[2]European Centre for Living Technology, 30123 Venice, Italy

Corresponding author: Kurt Stockinger (Kurt.Stockinger@zhaw.ch)

ABSTRACT Machine learning and quantum computing are both areas with considerable progress in recent years. The combination of these disciplines holds great promise for both research and practical applications. Recently there have also been many theoretical contributions of quantum machine learning algorithms with experiments performed on quantum simulators. However, most questions concerning the potential of machine learning on quantum computers are still unanswered such as *How well do current quantum machine learning algorithms work in practice? How do they compare with classical approaches?* Moreover, most experiments use different datasets and hence it is currently not possible to systematically compare different approaches. In this paper we analyze how quantum machine learning can be used for solving small, yet practical problems. In particular, we perform an experimental analysis of kernel-based quantum support vector machines and quantum neural networks. We evaluate these algorithm on 5 different datasets using different combinations of quantum feature maps. Our experimental results show that quantum support vector machines outperform their classical counterparts on average by 3 to 4% in accuracy both on a quantum simulator as well as on a real quantum computer. Moreover, quantum neural networks executed on a quantum computer further outperform quantum support vector machines on average by up to 5% and classical neural networks by 7%.

INDEX TERMS Machine learning, quantum computing, experimental evaluation.

I. INTRODUCTION

Hardly any other field of research in computer science has made such rapid progress in recent years as machine learning. It is used successfully in various areas both in research and in industry [18]. However, there are also limits and unsolved problems in practical applications due to the enormous computing resource requirements of large machine learning algorithms such as transformer-based language models [29]. Moreover, machine learning methods are often complex and based on large amounts of data. Therefore, depending on the task, the algorithms can become extremely computationally intensive.

A novel type of computer hardware, quantum computers, promises considerable speed-up so that these algorithms are useful for a broad class of users [5], [34]. Moreover, companies such as IBM and Amazon already provide public access to quantum computers via Python interfaces [1], [11]. This allows active research in quantum computing also for small to medium-sized research institutions or companies that do not have the computing resources of large corporations.

The field of *quantum machine learning* has gained considerable attention in the last years [4], [7], [10], [14]. However, it is still relatively unclear what kind of problems can be solved practically today and which ones remain only of theoretical nature.

In this paper we will perform an experimental evaluation of *quantum support vector machines (QSVM)* as well as *quantum neural networks (QNN)* and compare them against their classical counterparts. In our first set of experiments we will evaluate *kernel-based SVMs* [14]. Classical kernel-based SVMs have been studied well and have been widely applied.

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

However, the classical approaches suffer in situations where the feature space becomes large and the kernel functions become computationally expensive to estimate. These limitations can be overcome by using quantum algorithms that enable the exploitation of an exponentially large quantum state space through controllable entanglement and interference.

In addition, we study various implementations of QNNs based on different quantum circuits. One of the open research questions is how do *design optimal quantum circuits* both for QSVMs as well as QNNs such that the quantum algorithm shows the best learning behavior for practical machine learning problems. In order to address this question, we will perform an experimental evaluation of kernel-based QSVMs as well as QNNs for classification problems using five different datasets.

This paper makes the following *contributions*:

- We perform a detailed experimental evaluation of classical kernel-based SVMs and compare the accuracy against QSVMs running both on a quantum simulator and a real, publicly available quantum computer. We also compare the performance of classical neural networks against quantum neural networks.
- Our experimental evaluation on five different datasets shows that QSVMs outperform their classical counterparts on average by 3 to 4%. Moreover, QNNs further outperformed QSVMs by up to 5%.
- Further comparisons with classical neural networks demonstrate that QNNs also outperform those whilst using far fewer parameters, which has also been confirmed in comparable experiments [9].
- These results demonstrate that quantum computing can be successfully applied for small-scale machine learning problems in practice already today.

II. QUANTUM COMPUTING FOUNDATIONS

Quantum computing is a new computational paradigm based on the fundamental principles of quantum mechanics. The major concepts that can be leveraged for performing calculations are *superposition* and *entanglement* which basically means that quantum computation does to information what traditional quantum mechanics does to elementary particles and photons: it characterizes these fundamental entities by wave- and particle-like aspects. We will briefly explain these concepts below.

In a classical computer, information is stored in bits whose states can either be 0 or 1. In a quantum computer, the information is stored as *qubits* (quantum bits) to either represent the values 0 and 1 or a linear combination of both. Expressed in mathematical terms, a qubit is a vector of length 1 in a two-dimensional complex Hilbert space $\boldsymbol{H^2}$ (basis $|0\rangle$, $|1\rangle$). A general qubit $|q\rangle$ is given by $|q\rangle = c_0\,|0\rangle + c_1\,|1\rangle$ with $c_{0,1} \in \mathbb{C}$ and $|c_0|^2 + |c_1|^2 = 1$. The normalization means that a qubit is characterized by three real numbers ($c_n = a_n + ib_n$, $|c_n|^2 = |a_n|^2 + |b_n|^2$).

Classical bits can be combined into registers, which contain bit sequences. In contrast, the state of a quantum register of length n can be a linear combination of all possible bit sequences of length n. Expressed in mathematical terms, a quantum register is a state in the tensor product of n two-dimensional Hilbert spaces $\boldsymbol{H^{2^n}}$, i.e. a Hilbert space of dimension 2^n. The basis vectors can be written as:

$$|b_1\rangle \otimes \ldots \otimes |b_n\rangle = |b_1 b_2 \ldots b_n\rangle \tag{1}$$

with $b_k \in \{0, 1\}$. Here, $|b_1 b_2 \ldots b_n\rangle$ is a basis vector in $\boldsymbol{H^{2^n}}$. A general vector (or state, as it is called in quantum mechanics) is a linear combination of these basis states:

$$|Q\rangle \in \boldsymbol{H^{2^n}} : |Q\rangle = \sum_{b_k \in \{0,1\}} c_{b_0 b_1 \ldots b_n}\,|b_1 b_2 \ldots b_n\rangle \tag{2}$$

Quantum mechanics requires that the state $|Q\rangle$ is normalized:

$$\sum_{b_k \in \{0,1\}} |c_{b_0 b_1 \ldots b_n}|^2 = 1 \tag{3}$$

These linear combinations are called *superpositions*. There are two types of processes one can apply on such quantum registers:

- Quantum dynamics, which are *unitary transformations* (rotations and reflections) of $|Q\rangle$. These unitary transformations are reversible and fully deterministic.
- *Measurements*: These are projections combined with normalizations. For our purposes this means that a measurement M maps the state $|Q\rangle$ of a quantum register stochastically onto one of the basis states $|b_1 b_2 \ldots b_n\rangle$. The probability for this to happen is given by $|c_{b_0 b_1 \ldots b_n}|^2$. A measurement is irreversible and surjective (which implies that in a measurement, one looses information).

It helps to imagine a quantum computation as a series of unitary operations (true quantum operations) finalized with one measurement (more involved schemes are used, though). Importantly, rotating $|Q\rangle$ affects "all basis states at once", i.e. all basis vectors in Equation 2 are manipulated in parallel. Thus, a quantum computer is a highly parallel supercomputer.

The concept of entanglement implies that the combined state of qubits contains more information than the qubits have independently. This is easy and worthwhile to understand. We explain it in some detail, because a widespread misconception of quantum computing sees its advantages solely with respect to the mentioned "quantum parallelism".

We start with the observation that a single qubit is determined by three real numbers. A collection of n independent qubits is therefore characterized by $3n$ real values. A superposition is given by 2^n complex numbers being subject to the normalization condition in Equation 3. This implies that the state of $|Q\rangle$ is determined by $2 \cdot 2^n - 1$ real numbers. For the case of $n = 2$, this means that two independent qubits are characterized by six parameters, whereas the state of a quantum register of length $n = 2$ is determined by seven numbers.

Mathematically, this means that in general, the state of a quantum register $|Q\rangle$ cannot be written as a tensor product of two independent qubits $\left|q^A\right\rangle, \left|q^B\right\rangle$. The equation

$$\begin{aligned}\left|q^A\right\rangle \otimes \left|q^B\right\rangle &= (c_0^A\,|0\rangle + c_1^A\,|1\rangle) \otimes (c_0^B\,|0\rangle + c_1^B\,|1\rangle) \\ &= (c_0^A c_0^B\,|0\rangle \otimes |0\rangle + c_0^A c_1^B\,|0\rangle \otimes |1\rangle \\ &\quad + c_1^A c_0^B\,|1\rangle \otimes |0\rangle + c_1^A c_1^B\,|1\rangle \otimes |1\rangle \end{aligned} \tag{4}$$

describes the tensor product of two independent qubits. The state of a two-qubit register is given by

$$|Q\rangle = (c_{00}\,|00\rangle + c_{01}\,|01\rangle + c_{10}\,|10\rangle + c_{11}\,|11\rangle) \tag{5}$$

For a general choice of $c_{00}, c_{01}, c_{10}, c_{11}$ with $|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$ the equations $c_{00} = c_0^A c_0^B, c_{01} = c_0^A c_1^B, c_{10} = c_1^A c_0^B, c_{11} = c_1^A c_1^B$ cannot be solved under the normalization constraints $|c_0^A|^2 + |c_1^A|^2 = 1$ and $|c_0^B|^2 + |c_1^B|^2 = 1$.

Physically, there is in general no way to interpret the entangled state of a quantum register in terms of a collection of individual qubits.

In order to manipulate qubits, *quantum circuits* are used. These circuits are similar to their classical counterparts but they contain additional logical operators and gates. One of the gates is the Hadamard gate which brings qubits in a superposition.

Another important type of operator are the controlled Pauli-gates. Single qubits can be visualized as points on a two-dimensional sphere, the so called *Bloch-sphere*. The Bloch-sphere is embedded into a three-dimensional space with coordinate axis x, y, z. Note well that these coordinates have no direct relation to the actual physical space, but are primarily a consequence of a specific representation of qubits (a more detailed historical analysis of the origin of the Bloch-sphere would tell a somewhat different story, which is, however, not of relevance in the context here).

Controlled manipulation of qubits can then be understood as rotations around the x, y, z-axis, and consequently, these gates are also called *controlled X-, Y-* and *Z-gates*. As it turns out, since these rotations of one qubit depend on the state of another qubit, the application of such a controlled gate leads to quantum entanglement. Mathematically, all quantum gates can be considered as unitary matrix operations.

An example of a simple quantum circuit is given in Figure 1. The circuit consists of three qubits q_0, q_1 and q_2. First, all qubits are initialized with the ground state 0. Then, the Hadamard-gate is applied on qubit q_1, followed by controlled X-gate operation with qubit q_2 and a controlled X-gate operation between qubits q_0 and q_1, followed by a Hadamard-gate applied on qubit q_0. Finally, the qubits q_0 and q_1 are measured.

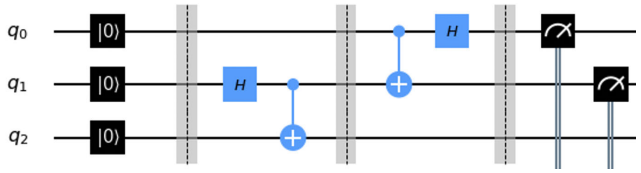


FIGURE 1. Example of a simple quantum circuit with three qubits. "H" refers to the Hadamard gate and "+" to a controlled X-gate. Finally, the states of qubits q_0 and q_1 are measured.

III. RELATED WORK

Over the last years, several different approaches of quantum machine learning algorithms [7] have been proposed. Some approaches are based on quantum support vector machines (QSVMs) [14], [24], while others are based on neural networks [2], [6], [12], [17], [21], [27], [33] and can be referred to as quantum neural networks (QNNs).

Let us start with analyzing QSVMs in more detail. [14] propose a variational quantum circuit to classify data similar to SVMs. This approach uses a variational circuit that generates a separating hyperplane in the quantum feature space. A further approach proposed by the same authors is called quantum kernel estimator, which is used to estimate the kernel function and optimize a classifier. To evaluate the approach, a synthetic data set is used that contains 20 data points per label. In our experiments we also use this data set.

One of the first study to demonstrate that quantum neural networks show an advantage over their classical counter parts is presented in [2]. The authors evaluate two different feature maps for the quantum neural network. One feature map is based on the circuit introduced in [14]. The second circuit uses parameterized RY-gates, which are followed by CNOT-gates that are applied between every pair of qubits in the circuit. Finally, another set of parameterized RY-gates are used. The QNN is evaluated both on a quantum simulator as well as on a real quantum computer using the Iris data set that we also use in our experiments.

A quanvolutional neural network architecture is proposed in [15]. The basic idea is to replace a convolutional filter of a CNN with a quanvolutional layer that transforms input data with a random quantum circuit. The approach is evaluated with image data on a quantum simulator. Our approach, however, is evaluated on various numerical data sets both on a quantum simulator as well as on real quantum hardware.

[8] introduce a hybrid classical-quantum approach called Quantum Short Long-Term Memory. The idea is to replace parts of classical RNN with a variational quantum circuit. The approach is evaluated on a quantum simulator but not on real quantum hardware.

In this paper we evaluate existing approaches based on QSVMs and QNNs. Previous algorithms have mostly been evaluated either only on a quantum simulator or on a single data set. Hence, comparability of the algorithms as well as generalizability of the approaches to other data sets has not been demonstrated in depth. In our paper, we evaluate various quantum machine learning approaches on 5 different data sets both on a quantum simulator as well as on real quantum hardware.

The major question we study in this paper is how well these algorithms perform on small, yet real machine

learning problems using publicly available quantum hardware.

IV. MACHINE LEARNING APPROACHES

A. KERNEL-BASED SVMs: CLASSICAL APPROACH

A *feature function* $\phi(\vec{x})$ is a mapping of a data point $\vec{x}$ into *feature space* of higher dimension. This is advantageous for classification because it opens up more possibilities for a hyperplane to separate data point of different classes.

The so-called *kernel trick* allows re-writing of a linear decision function used by SVMs in terms of a dot product between data points. In combination with a feature function, it can be further substituted with a *kernel function* $k(\vec{x}, \vec{x}^{(i)}) = \phi(\vec{x})^T \cdot \phi(\vec{x}^{(i)})$, for a given training data point $\vec{x}^{(i)}$ and a data point $\vec{x}$ for which the decision is made [13]. The decision function in its final form $f(\vec{x}) = b + \sum_i \alpha_i k(\vec{x}, \vec{x}^{(i)})$ introduces a shortcut to the explicit calculation of the dot product between feature vectors, which can be of infinite dimension. Furthermore the resulting function is linear in the feature space.

The part $\sum_i \alpha_i k(\vec{x}, \vec{x}^{(i)})$ of the function is called *kernel matrix* and represents the similarity values between each training data point.

B. KERNEL-BASED SVMs: QUANTUM APPROACH

Let us now discuss how classification can be implemented on a quantum computer. In principle, we need to following steps:

- Transform the classical data points into quantum data points with a quantum circuit.
- Use a parameterized quantum circuit to classify the data.
- Measure the output.
- Send the results of the quantum kernel to a classical SVM for final classification.

These steps comprise a so-called *variational quantum classifier* [31] leveraging parameterized quantum circuits. Since current quantum computers are still quite error-prone, a common approach is to implement one part of the end-to-end process on a quantum computer and the remaining parts on a classical computer. In particular, [14] suggest a *quantum kernel estimator*, where the kernel function is implemented as a quantum kernel, i.e. a quantum circuit, which translates classical data into quantum states via a *quantum feature map*, and then builds the inner product of these quantum states.

The inner product is used for further processing by the classical SVM. As a final step, the classification is performed by a kernel-based SVM on a conventional computer using the calculated kernel. In summary, the calculation of the kernel matrix is performed by a quantum algorithm, whereas the classical SVM algorithm is executed on a conventional computer.

Let us describe the QSVM approach more formally. According to Thomsen [30], using an already classified data point $\vec{x}^{(i)}$ and a to be classified data point $\vec{x}$, the corresponding decision function uses the kernel function to classify $\vec{x}$,

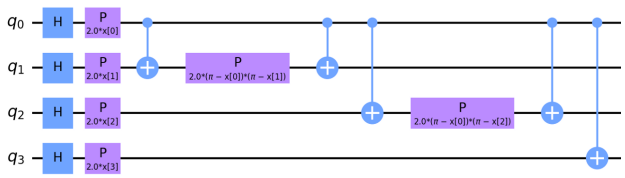


FIGURE 2. Example of a quantum kernel based on the Pauli-feature-map. For simplicity, the figure only shows parts of the circuit.

as shown in Equation 6

$$QSVM(\vec{x}) = sign\left(\sum_{i=1}^{M} a_i y_i \left\langle \phi(\vec{x}^{(i)}), \phi(\vec{x}) \right\rangle_{\mathcal{H}} + b\right) \quad (6)$$

where a and b are training parameters and M refers to the size of the data set. The quantum analogue kernel function equation – with an exponentially large space of density matrices $\mathcal{S}(2^q)$ spanned across q qubits as the feature space – can be seen in Equation 7 [30].

$$\begin{aligned} \psi : \mathbb{R}^S &\to \mathcal{S}(2^q) \\ \vec{x} &\mapsto \left|\psi(\vec{x}^{(i)})\right\rangle\left\langle\psi(\vec{x})\right| \\ \to k(\vec{x}, \vec{x}^{(i)}) &= \left|\left\langle\psi(\vec{x}^{(i)})\middle|\psi(\vec{x})\right\rangle\right|^2 \end{aligned} \quad (7)$$

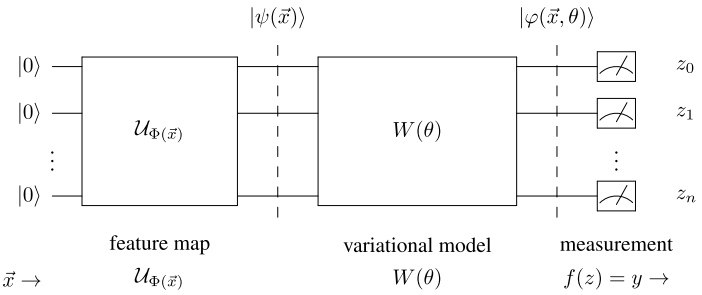
As previously stated, the quantum circuit, which performs the transformation into the quantum space, is called *quantum feature map*. Typical quantum feature maps are the Z-feature-map, the ZZ-feature-map and the Pauli-feature-map [14].

An example of the Pauli-feature-map, which is the most generic feature map, is shown in Figure 2. It consists of two different quantum gates, namely the Hadamard gate, which puts qubits in superposition, and a parameterized P-gate (phase gate). In addition, we can see the controlled X-gate ("+") which enables entanglement between qubits.

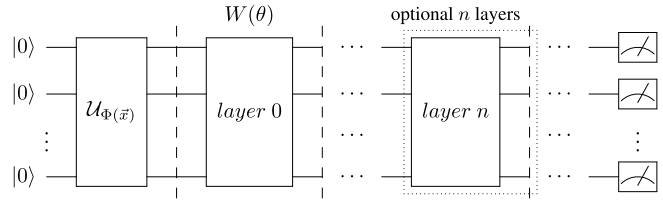
The circuit can also be stacked and thus made wider in order to design even more complex feature maps resulting in a quantum circuit with a larger depth. However, due to the limitations of current quantum devices, larger quantum circuits often lead to a higher error rate. Hence, designing optimal quantum kernels for SVMs is still an unsolved research problem. The goal of this paper is to evaluate various feature maps for solving small, yet practical machine learning problems using QSVMs.

C. QUANTUM NEURAL NETWORK

The design of the quantum neural network is inspired by previous work of Havlicek et al. [14] and Thomsen [30]. The general architecture of the quantum circuit is shown in Figure 3a and consists of three parts. The first part is the *feature map* $\mathcal{U}_{\Phi(\vec{x})}$ which is used to encode the input features of the used dataset into quantum states. The second part is the *variational model* $W(\theta)$ which evolves the quantum states of the system using trainable parameters θ. The final layer consists of the measurement of the final states.



(a) A fixed input $\vec{x}$ is encoded into the quantum state by applying $\mathcal{U}_{\Phi(\vec{x})}$ to the reference state $|0\rangle^n$ for n qubits: $\mathcal{U}_{\Phi(\vec{x})}|0\rangle^n = |\psi(\vec{x})\rangle$. The state then is evolved via the variational unitary $W(\theta)$ with trainable parameters θ: $W(\theta)|\psi(\vec{x})\rangle = |\varphi(\vec{x},\theta)\rangle$. The resulting bit string $z \in \{0,1\}^n$ is post-processed to an associated label in $y := f(z)$ with y as output.



(b) The variational model $W(\theta)$ can be repeated n times which can thus be considered as layers of the QNN.

FIGURE 3. Architecture of the variational quantum circuits used in the QNN experiments.

Figure 3b shows that the variational model can be repeated n times, which is similar to the layers of a classical neural network. The larger the quantum circuit, the better a function can be approximated and hence the better the generalization of the machine learning algorithm should be. At the same time, current quantum hardware is limited in its size and stability. Available systems do not allow for the creation of longer circuits with repeated, variational models. At the same time the length of a circuit should be minimized - noise affecting the quantum system during calculations leads to instabilities and influences the resulting measurements, which can falsify results. With this in mind, the circuits in this paper are limited to using a single instance of the variational model.

The parameters θ of the variational model are optimized using classical optimizers leveraging classical hardware.

Note that there is a fundamental difference between the QSVM we discussed previously and the QNN we described here. In the case of the QSVM, only the kernel is implemented using a quantum circuit while the SVM itself is implemented classically. As for the QNN, the whole neural network is implemented using a quantum circuit and only the optimization of the parameters is implemented classically.

We will now describe the implementation of the feature map and the variational model in more detail.

1) FEATURE MAP

The main goal of the feature map is to encode classical features of our dataset into the Hilbert space $\mathcal{H}$ the quantum system acts in. We apply the circuit $\mathcal{U}_{\Phi(\vec{x})}$ to the zero state $|0\rangle$, which defines the feature map as described in Equation 8

$$|\psi(\vec{x})\rangle := \mathcal{U}_{\Phi(\vec{x})}|0\rangle \tag{8}$$

where $\vec{x}$ is defined according to Equation 7 previously introduced in Section IV-B.

Among a multitude of embedding techniques [25], we have chosen *angle encoding* to encode the classical data into quantum states. Whilst angle encoding is not optimal as it requires n qubits to represent n-dimensional data, it is efficient regarding operations and directly useful for processing data in quantum neural networks [19]. Weigold et al. [32] state that only single-qubit rotations are needed for the state preparation routine, which is highly efficient and can be done in parallel for each qubit.

Figure 4 shows a circuit with one qubit per feature. For instance, qubit q_0 represents feature x_0 which is encoded with a rotation around the y-axis where the angle is proportional to the value of feature x_0.

2) VARIATIONAL MODEL

After the classical features are encoded as quantum states, these quantum states can be further evolved in the variational model according to Equation 9.

$$|\psi(\vec{x},\theta)\rangle := W(\theta)\mathcal{U}_{\Phi(\vec{x})}|0\rangle \tag{9}$$

The trainable weights are embedded in the variational model $W(\theta)$ which can be grouped into n layers, where each layer consists of RY-, RX-, and RZ-rotation gates.

An example of a variational model with three parameters is shown in Figure 5. Note that first qubit q_0 is rotated by angle $2\theta_1$ around the y-axis. Next, qubit q_0 and qubit q_1 are entangled with a controlled R_y-gate with the angle parameter

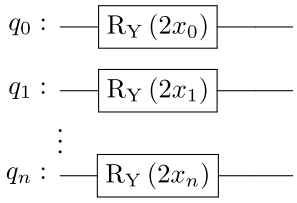


FIGURE 4. Quantum circuit demonstrating angle encoding with RY-gates.

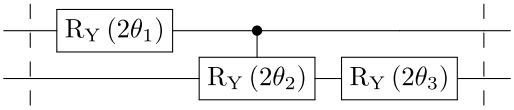


FIGURE 5. Variational model with three parameters.

$2\theta_2$ followed by a rotation around the y-axis with the angle parameter $2\theta_3$.

3) DECISION FUNCTION

Next, the resulting state of Equation 9 needs to be measured. Since we use our quantum circuit as a binary classifier, a bitstring $z \in \{0, 1\}^q$ is calculated which is associated with a class membership via the following Boolean function.

$$f : \{0, 1\}^q \to \{-1, +1\} \\ z \mapsto \tilde{y} \tag{10}$$

The classification is re-run multiple times (R shots) where R is the number of re-runs or shots. Thus, the resulting measurement outcome z is probabilistic and we need to assign the label to the bitstring with the largest probability. Hence the probability of measuring either label $y \in \{+1, 1\}$ is given by

$$p_y = \frac{1}{2}(1 + y \langle\psi(\vec{x})| W(\theta)^\dagger \mathcal{F} W(\theta) |\psi(\vec{x})\rangle) \tag{11}$$

where $\mathcal{F}$ is a diagonal operator

$$\mathcal{F} = \sum_{z \in \{0,1\}^n} f(z) |z\rangle \langle z| \tag{12}$$

Since $\mathcal{F}$ only has eigenvalues of -1 or $+1$, the expectation value is as follows:

$$QNN_\theta(\vec{x}) = \langle\psi(\vec{x})| W(\theta)^\dagger \mathcal{F} W(\theta) |\psi(\vec{x})\rangle \in [-1, +1] \tag{13}$$

V. EXPERIMENTS

In our first set of experiments we evaluate the performance of classical kernel-based support vector machines and compare them against QSVMs. First, we execute the QSVMs on qasm_simulator, a Python-based quantum simulator of IBM Qiskit [3] accessed via BasicAer.[1] Afterwards, we execute the experiments on ibmq_belem, a real quantum system providing 5 qubits [16], accessed via IBMQ.[2]

In our second set of experiments we compare the accuracy of classical neural networks against quantum neural networks.

[1]https://qiskit.org/documentation/apidoc/providers_basicaer.html
[2]https://qiskit.org/documentation/apidoc/ibmq_provider.html

TABLE 1. Characteristics of the five datasets used for our experiments.

Dataset	#Features	#Records	#Classes
Iris	4	100	2
Rain	5	100	2
Vlds	5	100	2
Custom	2	100	2
Adhoc	3	100	2

The major questions we address with these experiments are as follows:

- *Which quantum circuit yields the best performance for a given dataset?*
- *Can we establish a clear strategy for designing quantum circuits?*
- *Does the quantum implementation of the algorithm have an advantage over the classical counterpart?*

A. DATASETS

We will now describe the datasets that we used for our experiments. In particular, we used five datasets with varying degrees of difficulty, which was estimated from the order of the separating hyperplane in the origin space. Each of these datasets has one hundred data points and two classes containing the same number of data points.

The chosen encoding strategy assumes one qubit per feature. Since our quantum computer provides a maximum of five qubits, we have reduced the number of features to a maximum of five, if necessary. For training and testing we performed an 80:20 percent split. Moreover, we performed a 10-fold cross validation for all experiments and report on the average results. An overview of the datasets is given in Table 1.[3]

Iris dataset. This widely used flower dataset was loaded via the Python library scikit-learn.[4] For the experiment, the data points were selected from the Iris-Setosa and Iris-Virginica classes. A data point has four numerical features.

Rain dataset. This dataset is taken from kaggle.com[5] and contains about ten years of daily weather observations from many locations in Australia. The incomplete data entries were removed, and the following five features were selected for this purpose: MinTemp, Humidity9am, WindSpeed3pm, Pressure9am, WindDir9am. The attribute RainTomorrow serves as a class label. Its categorical values No and Yes were mapped to the numbers 0 and 1, respectively.

Vlds dataset. This dataset was generated using a dataset generator provided by scikit-learn.[6] The characteristics of the features are shown in Figure 6.

[3]Note that currently we use relatively small datasets consisting of 100 records. One of the reason is that we use a publicly available quantum computer with access limitations where users are put in a task queue. When a single user uses too many resources, the access is blocked.
[4]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html
[5]https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
[6]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_multilabel_classification.html?highlight=make_multilabel_classification

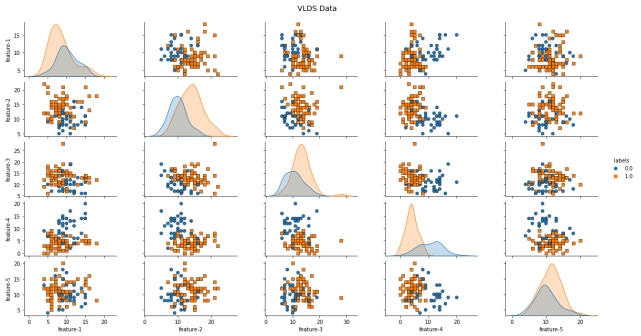


FIGURE 6. From the pairwise bivariate distributions of the Vlds dataset, it is apparent that a higher order function is required to separate them, indicating the difficulty of the dataset.

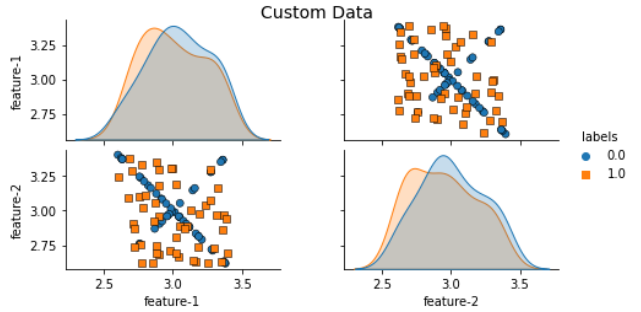


FIGURE 7. From the pairwise bivariate distributions of the custom dataset, it is apparent that a higher order function is required to separate them, indicating the difficulty of the dataset.

Custom dataset. The dataset consists of data points with two features, generated using the function numpy.random.default_rng.[7] In a 2D representation, the data points are part of a square. The points on the diagonals are labelled as 0.0 (see Figure 7).

Adhoc dataset. This dataset is artificially generated and described in [14]. It provides a complete classification of data points using the feature map configurations of the quantum kernel estimator approach chosen in [14]. Qiskit provides an Adhoc dataset generator,[8] allowing the generation of data points with three features. The characteristics of the features are shown in Figure 8.

B. CLASSICAL KERNEL-BASED SUPPORT VECTOR MACHINES

We first evaluate the accuracy of classical kernel-based SVMs provided by sklearn.svm.SVC with default hyperparameter configurations, using four different kernels, namely linear, polynomial, radial basis function and sigmoid. We have used these kernels of different complexity to be able to adapt to the datasets, that are also of different complexity.

The results are shown in Table 2. As we can see, for the Iris dataset, all kernels have a perfect accuracy score of 1.00. For the Rain dataset, the rbf kernel performs best with an accuracy of 0.77. For the Vlds dataset, all kernels behave similarly with a slight advantage for the linear kernel. For the custom dataset, again the rbf kernel performs best. Finally, for the Adhoc dataset, which is the most complex one, the linear kernel performs best with an accuracy score of 0.56 followed by sigmoid and rbf. In general, the rbf kernel appears to be the most robust one across all five datasets.

[7] https://numpy.org/doc/stable/reference/random/generator.html

[8] https://qiskit.org/documentation/stable/0.26/_modules/qiskit/ml/datasets/ ad_hoc.html#ad_hoc_data

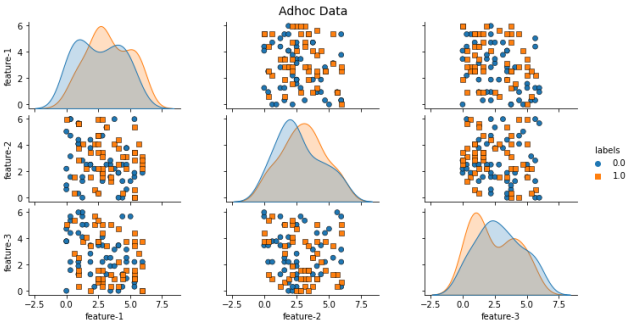


FIGURE 8. Pairwise bivariate distribution of the ad hoc dataset indicates a higher order partition function, indicating the difficulty of the dataset.

C. QUANTUM SUPPORT VECTOR MACHINES ON QUANTUM SIMULATOR

Let us now analyze the performance of QSVMs on a quantum simulator. Figure 9 shows the accuracy of QSVMs using three different feature maps and four different entanglement strategies (none, linear, circular, full). To analyze the effect of the circuit depth on the accuracy, we set the circuit depth to either 1, 2, 4 or 8.

We first consider the results for the Iris dataset. Figure 9a shows that with the Z-feature-map of depth 1 and 4, as well as with the ZZ-feature-map and the Pauli-feature-map of depth 1 the accuracy of 100% can be achieved. For all other feature maps with a depth above 2 we can observe a relatively high variance which might be due to the characteristics of the different data samples due to the 10-fold cross validation. We also notice that introducing entanglement harms the performance of the algorithm.

For the Rain dataset 9b we can clearly see that the Z-feature-map of depth 1 outperforms all other feature maps. The same pattern is observed with the Vlds dataset.

For the Custom and the Adhoc datasets, which are considered to be the most complex datasets, when can again see that the Z-feature-map performs best. Moreover, it can be recognized, that increasing the depth slightly improves the accuracy.

In summary, we can observe that the Z-feature-map performs best across all five datasets. We can also see that a greater depth of the feature map circuits can have a positive effect on the accuracy for more complex datasets. Finally, the more complex feature maps ZZ-feature-map and Pauli-feature-map have a negative impact on the accuracy. The latter also suggests that these algorithms cannot take advantage of entanglement. Hence, additional studies are required to understand these phenomena in more detail.

FIGURE 9. Accuracy of different quantum Support Vector Machines using three different feature maps (quantum kernels) with four different entanglement strategies on a quantum simulator for five different datasets.

TABLE 2. Accuracy of classical Support Vector Machines with four different kernel functions. The best results are marked in bold.

Dataset	Kernel Function	Accuracy
Iris	linear	1.00
	poly	1.00
	rbf	1.00
	sigmoid	1.00
Rain	linear	0.70
	poly	0.75
	rbf	**0.77**
	sigmoid	0.64
Vlds	linear	**0.89**
	poly	0.87
	rbf	0.86
	sigmoid	0.88
Custom	linear	0.48
	poly	0.48
	rbf	**0.64**
	sigmoid	0.59
Adhoc	linear	**0.56**
	poly	0.50
	rbf	0.54
	sigmoid	0.55

TABLE 3. Comparison of classical kernel-based support vector machines with quantum support vector machines on a quantum simulator as well as on a real quantum computer. The table shows the accuracy of the best approach for each of the three algorithm types.

Dataset	Classical SVM	QSVM (Quantum Simulator)	QSVM (Quantum Computer)
Iris	1.00	1.00	1.00
Rain	0.77	0.77	0.70
Vlds	0.89	0.86	0.90
Custom	0.64	0.76	0.76
Adhoc	0.56	0.65	0.60
Average	**0.77**	**0.81**	**0.80**

D. QUANTUM SUPPORT VECTOR MACHINES ON QUANTUM COMPUTER

Let us now evaluate the performance of quantum support vector machines on a real, publicly available quantum computer. The major question is if the algorithms still perform well on a quantum device or if the failure rates of the underlying quantum computer render these types of quantum machine learning algorithms impractical.

Our experimental results on a real quantum computer showed similar results to the ones on a quantum simulator. These results are extremely promising since current quantum computers are still very error-prone especially for circuits with a large number of quantum gates. Hence, being able to run quantum machine learning algorithms on real quantum computers that outperform their classical counterparts is a very promising step in quantum space.

In Table 3 we compare the best results of the classical kernel-based SVMs with the QSVMs on the quantum simulator as well as on the real quantum computer. As we can see, the QSVMs have a higher average accuracy over all five datasets than the classical counterparts and thus outperform them by 4% and 3%, respectively.

E. QUANTUM NEURAL NETWORKS

We will now evaluate the performance of quantum neural networks. Recall that we use a hybrid approach where the neural network is implemented as a variational quantum circuit and the optimizer is implemented using classical hardware.

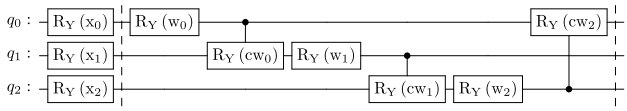


FIGURE 10. QNN circuit variant with 3 qubits and 1 layer used in our experiments. Referred to as *q_circuit_01*.

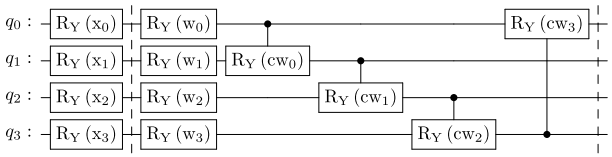


FIGURE 11. QNN circuit with 4 qubits and 1 layer used in our experiments. Referred to as *q_circuit_02*.

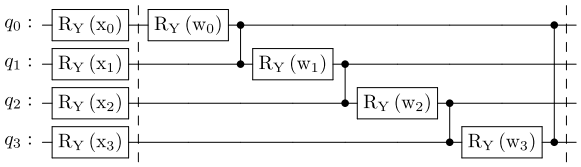


FIGURE 12. QNN circuit with 4 qubits and 1 layer used in our experiments. The qubit or layer count may vary depending on the input feature count or layer count setting. Referred to as *q_circuit_03*.

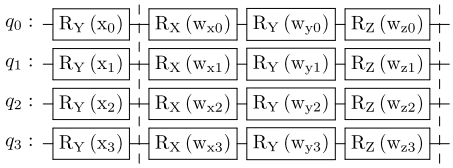


FIGURE 13. QNN circuit with 4 qubits and 1 layer used in our experiments. Referred to as *q_circuit_04*.

In our experiments we use the following 5 quantum circuits. To increase expressiveness in different ways [28], all circuits use at least one or a combination of the *RY*, *RX*, *RZ* gates. There is only one circuit without entanglement which is *q_circuit_04*. All other circuits use entanglement by including either a *CX*, *CZ* or a *CRZ* gate in their variational model inspired by [26].

1) *q_circuit_01*

The circuit in Figure 10 is built using circular entanglement with RY-gates followed by parameterized entangled CRY-gates consisting of 1 layer.

2) *q_circuit_02*

The circuit in Figure 11 is built using RY-gates followed by circular entangled parameterized CRY-gates depicted with 1 layer.

3) *q_circuit_03*

The circuit in Figure 12 is built using circular entanglement with RY-gates followed by entangled CZ-gates and is depicted with 1 layer.

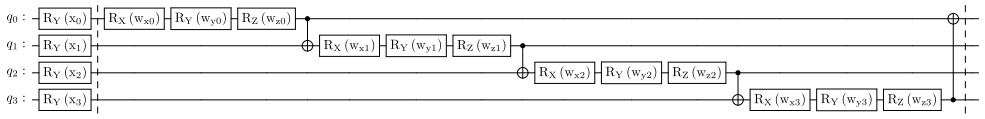


FIGURE 14. QNN circuit with 4 qubits and 1 layer used in our experiments. Referred to as *q_circuit_05*.

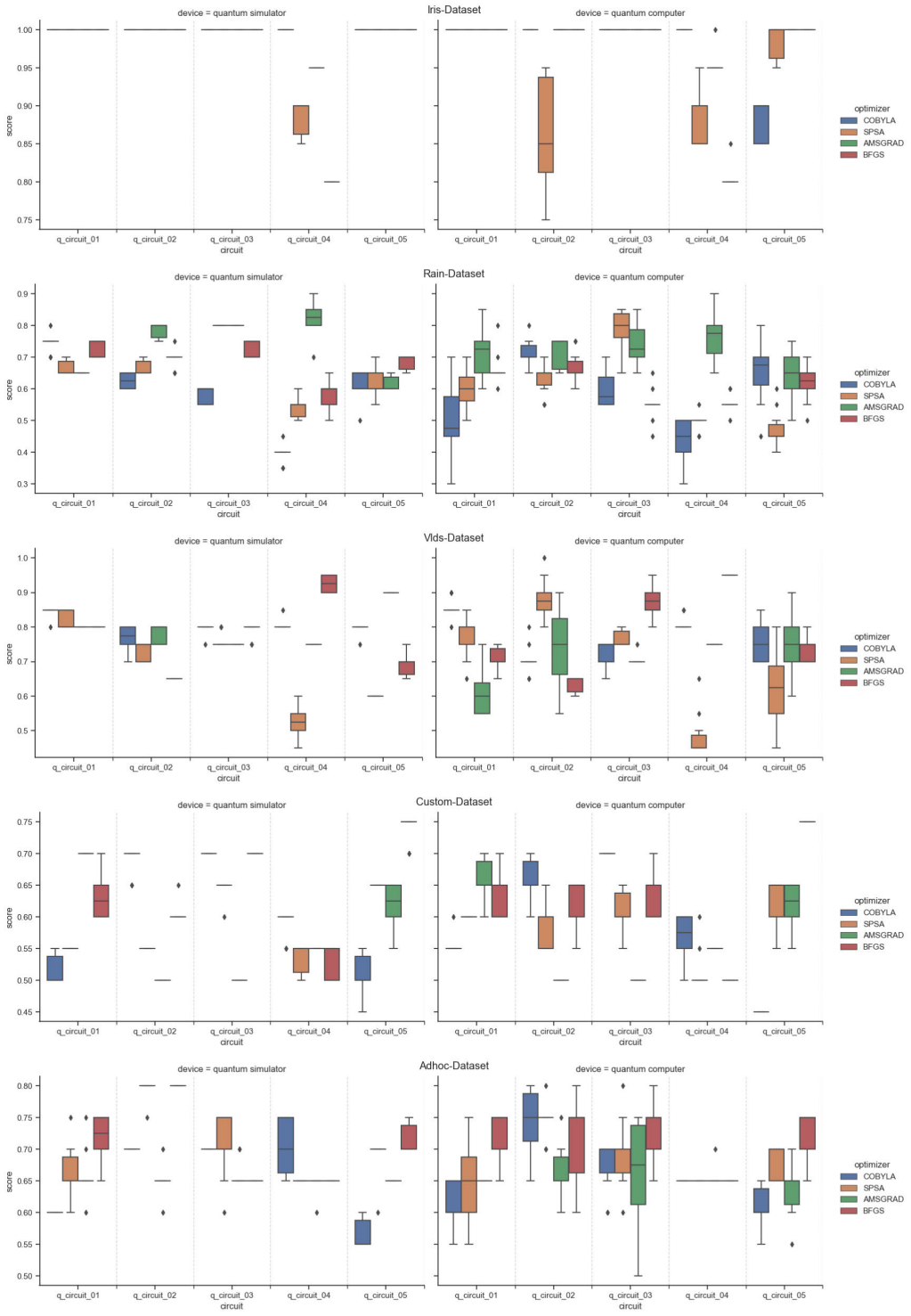


FIGURE 15. Accuracy comparison of 10 runs on a quantum simulator and a quantum computer over 5 datasets using 5 different quantum circuits and 4 different optimizers.

TABLE 4. Comparison of classical neural networks with quantum neural networks on a quantum simulator as well as on a real quantum computer. The table shows the accuracy of the best approach for each of the three algorithm types.

Dataset	Classical NN	QNN (Quantum Simulator)	QNN (Quantum Computer)
Iris	1.00	1.00	1.00
Rain	0.70	0.83	0.79
Vlds	0.94	0.93	0.95
Custom	0.64	0.74	0.75
Adhoc	0.61	0.80	0.75
Average	**0.78**	**0.86**	**0.85**

4) *q_circuit_04*

The circuit in Figure 13 is built using RX-gates followed by RY-gates and final RZ-gates. This circuit is without entanglement.

5) *q_circuit_05*

The circuit in Figure 14 is built using circular entanglement with RX-gates followed by RY-gates and final RZ-gates entangled by CX-gates and is depicted with only 1 layer.

6) CLASSICAL OPTIMIZERS

Inspired by the work of Pellow-Jarman et al. [23], we selected the same four optimizers: AMSGRAD, SPSA, BFGS and COBYLA. For all optimizers we tuned the parameter settings ranging from 100 to 1500 iterations.

7) RESULTS OF NEURAL NETWORKS

For the classical approach we have implemented various fully-connected neural networks with 1, 2 and 3 hidden layers using PyTorch [22]. These neural networks were then passed to Ray Tune [20], which is used to facilitate the search for good hyperparameters when optimizing neural networks. The best validation accuracy was selected out of 10 runs with the best hyperparameters. The input data was normalized in the same way it was for the quantum neural network, as to eliminate the normalization as a deciding factor. As we can see in Table 4, the average accuracy of the best classical neural networks over all 5 datasets is 78%, which is similar to the performance of the classical SVM shown in Table 3.

We will now evaluate the performance of the quantum neural networks. Figure 15 shows the accuracy of the quantum neural networks with different classical optimizers on 5 datasets. Let us first analyze the performance on the Iris dataset. On the simulator we can see that except for QNNs using the SPSA-optimizers, all others receive a perfect score of 100% accuracy. On the quantum computer we can see a similar behavior.

For the rain dataset, the highest accuracy of 82% can be achieved with the AMSGRAD-optimizer on the quantum simulator. On the quantum hardware, the SPSA-optimizer shows a slight advantage followed by AMSGRAD.

For the Vlds dataset, the BFGS-optimizer shows the highest accuracy. For the custom dataset and the adhoc dataset the winner is COBYLA. In short, for all the different experiments there is no clear winning optimizer.

When looking at the quantum circuits, it also turns out that there is no clear winner and there is a relatively high variance between the different circuits. The reason might be due to the variations of the dataset and the relatively small number of data records.

Table 4 shows the performance of the best combination of quantum circuits and optimizers per dataset. On average, the accuracy over all 5 datasets is 85.8% on the quantum simulator and 84.7% on the quantum computer. The results of the QNN are 5% better than the results of the QSVM which demonstrates the advantage of quantum implementations over a hybrid quantum-classical implementation. Moreover, the quantum neural network executed on the quantum computer outperforms the classical neural network by 7% - even though the classical neural network is vastly more complex. In the case of the vlds dataset, hyperparameter optimization resulted in a neural network with 69,402 parameters, whereas as the biggest quantum neural network has 15 parameters.

VI. CONCLUSION

In this paper we performed a detailed experimental evaluation of quantum support vector machines and quantum neural networks. Our experimental evaluation showed that QSVMs outperform their classical counterparts on average by 3 to 4% in terms of accuracy. We could also show that the quantum neural networks further outperformed the QSVMs by up to 5%.

Even though our experiments were only performed on relatively small datasets, these results demonstrate that quantum computing can be successfully applied for small-scale machine learning problems in practice already today. Given the tremendous progress in the development of quantum hardware, we expect that also larger problem sizes can be tackled in the near future. Whilst only usable for problems of a limited size, they outperform classical solutions on the same problems, whilst being, comparatively, less complex.

Our current experiments showed that the best quantum kernel is based on the Z-feature-map which does not use quantum entanglement. One of the open research questions is how to design quantum circuits such that they can take advantage of entanglement and thus harness of the full power of quantum computing. Another open research question is to find out how the analyzed algorithms perform on larger datasets. Larger, less error-prone quantum hardware might give more insights.

REFERENCES

[1] *Amazon Bracket*. Accessed: Jan. 2022. [Online]. Available: https://aws.amazon.com/braket/
[2] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Comput. Sci.*, vol. 1, no. 6, pp. 403–409, 2021.
[3] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernàndez, J. Carballo-Franquis, A. Chen, C. F. Chen, and J. M. Chow. (2019). *Qiskit: An Open-Source Framework for Quantum Computing*. Accessed: Mar. 16, 2019. [Online]. Available: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Qiskit%3A+An+Open-source+Framework+for+Quantum+Computing&btnG=

[4] S. Arunachalam and R. De Wolf, "Guest column: A survey of quantum learning theory," *ACM SIGACT News*, vol. 48, no. 2, pp. 41–67, Jun. 2017.
[5] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, and B. Burkett, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
[6] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Sci. Technol.*, vol. 4, no. 4, Nov. 2019, Art. no. 043001.
[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, Sep. 2017.
[8] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, "Quantum long short-term memory," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 8622–8626.
[9] E. A. Cherrat, I. Kerenidis, N. Mathur, J. Landman, M. Strahm, and Y. Y. Li, "Quantum vision transformers," 2022, *arXiv:2209.08167*.
[10] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, "Quantum machine learning: A classical perspective," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 474, no. 2209, 2018, Art. no. 20170551.
[11] A. Cross, "The IBM Q experience and QISKit open-source quantum computing software," in *Proc. APS March Meeting Abstr.*, 2018.
[12] Y. Du, M.-H. Hsieh, T. Liu, S. You, and D. Tao, "Learnability of quantum neural networks," *PRX Quantum*, vol. 2, no. 4, Nov. 2021, Art. no. 040337.
[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[14] V. Havlicek, A. D. Corcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.
[15] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," *Quantum Mach. Intell.*, vol. 2, no. 1, pp. 1–9, Jun. 2020.
[16] IBM Quantum Team. (2022). *IBMQ_melbourne V2.3.24*. Accessed: Jan. 2022. [Online]. Available: https://quantum-computing.ibm.com
[17] S. K. Jeswal and S. Chakraverty, "Recent developments and applications in quantum neural network: A review," *Arch. Comput. Methods Eng.*, vol. 26, no. 4, pp. 793–807, Sep. 2019.
[18] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015.
[19] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Sci. Technol.*, vol. 5, no. 4, Oct. 2020, Art. no. 044007.
[20] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," 2018, *arXiv:1807.05118*.
[21] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Commun.*, vol. 9, no. 1, pp. 1–6, Nov. 2018.
[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[23] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione, "A comparison of various classical optimizers for a variational quantum linear solver," *Quantum Inf. Process.*, vol. 20, no. 6, p. 202, Jun. 2021.
[24] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Phys. Rev. Lett.*, vol. 113, no. 13, Sep. 2014, Art. no. 130503.
[25] M. Schuld, "Supervised quantum machine learning models are kernel methods," 2021, *arXiv:2101.11020*.
[26] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Phys. Rev. A, Gen. Phys.*, vol. 101, no. 3, Mar. 2020, Art. no. 032308.
[27] M. Schuld, I. Sinayskiy, and F. Petruccione, "The quest for a quantum neural network," *Quantum Inf. Process.*, vol. 13, no. 11, pp. 2567–2586, Nov. 2014.
[28] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Adv. Quantum Technol.*, vol. 2, no. 12, Dec. 2019, Art. no. 1900070.
[29] D. So, Q. Le, and C. Liang, "The evolved transformer," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5877–5886.
[30] A. Thomsen, "Comparing quantum neural networks and quantum support vector machines," M.S. thesis, Inst. Theor. Phys., Zurich, Switzerland, Jun. 2021.
[31] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Phys. Rev. A, Gen. Phys.*, vol. 92, no. 4, Oct. 2015, Art. no. 042303.
[32] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Encoding patterns for quantum algorithms," *IET Quantum Commun.*, vol. 2, no. 4, pp. 141–152, Dec. 2021.
[33] C. Zhao and X.-S. Gao, "Analyzing the barren plateau phenomenon in training quantum neural networks with the ZX-calculus," *Quantum*, vol. 5, p. 466, Jun. 2021.
[34] H.-S. Zhong, H. Wang, Y. H. Deng, M. C. Chen, L. C. Peng, Y. H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, and P. Hu, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020.