Quantum aspects of variational quantum circuit-based models

MSc Thesis

Supervisors: Dr. Vedran Dunjko & Dr. Jordi Tura

First examiner: Prof. Dr. Anastasia Borschevsky

Second examiner: Dr. Ir. Gerco Onderwater

Marius van Laar

July 2022

Abstract

Variational Quantum Circuits (VQCs) are a class of Quantum Machine Learning (ML) methods which have successfully been implemented on near-term devices. However, VQCs do not benefit from any rigorous advantages over classical methods. Where previous work has focused on establishing theoretical bounds in capacity or generalization error, we probe the change in learning performance as the classical computational hardness of the model increases. First we develop three simulation methods which offer systematic control of the computation hardness; two for the simulation of Matchgate circuits with few SWAP gates, and one based on the Gate Cutting technique. We introduce a VQC model based on the latter technique, and find the computational hardness has no impact on the learning performance for a range of real-world and quantum datasets.

Contents

Chapter 1

Introduction

Quantum computers are devices based on the principles of quantum mechanics, and as such offer fundamentally different methods of computation to classical devices. Research into algorithms for such quantum devices have shown advantages for certain problems over classical computers, through superior scaling as the problem size increases. Examples include solving a linear system of equations through the HHL algorithm [1], finding discrete logarithms and prime factorization [2] and solving complex quantum many-body problems in quantum chemistry [3]. These algorithms rely on the unique aspects of quantum devices to produce complex data distributions or perform particular operations which are difficult for a classical device to produce or process.

Classical Machine Learning methods are capable of not only recognizing correlations and patterns in data distributions invisible to the human eye, but also generating new data with the same correlations, patterns and statistics. If quantum devices offer access to classically difficult processing capabilities and data distributions, it motivates research into how Quantum computing and Machine Learning can benefit each other. One goal might be finding algorithms and methods which are capable of learning, and consequently reproducing, complex quantum data distributions which are computationally difficult to sample from using classical methods.

Research into this interdisciplinary field, dubbed Quantum Machine Learning (QML), has already generated many new (quantum) algorithms and ideas, many of which are inspired by or generalizations of established classical methods [4]. A simple yet effective approach is to use a quantum circuit as a model, which contain gates with free parameters that introduce the variational component necessary to create a continuous family of functions. These free parameters can be optimized classically for a given problem, yielding a hybrid quantum-classical algorithm which is broadly referred to as a Variational Quantum Algorithm (VQA) [5]. This approach is already powerful enough to yield a universal function approximator with a single qubit, at the expense of a large number of gates [6]. Whilst this proves the relevance of quantum models for machine learning, a single qubit is still classically simulable.

What is it about these QML algorithms that gives them an advantage over classical methods? Advantages focused on improvements in time or query complexity are often provable using theoretical arguments at the cost of stringent quantum hardware requirements [7]. Consequently nearly all methods supported by strong evidence of classical-quantum separations are unfeasible to run on near-term devices, characterized by low gate fidelity, qubit count (up to 100 qubits) and short decoherence times. These near-term devices are referred to as Noisy Intermediate-Scale Quantum (NISQ) computers [8]. For example, one caveat to the HHL algorithm is the requirement to encode N bits of information in $\log_2(N)$ qubits, using a process called amplitude encoding. It is known that this process generally requires exponentially many gates [9] and hence is unfeasible for many near-term devices due to the accumulation of noise. Furthermore, these advantages are in the time or query complexity and therefore do not imply any improvement in performance on real world datasets. It remains a challenge to properly contrast and compare classical and quantum machine learning methods suitable for NISQ devices within the same framework [10], and identify practical settings where quantum methods have a tangible advantage over classical methods.

Our pragmatic approach to analysing the difference between quantum and classical methods is to consider how difficult the quantum method is to classically simulate. The direct classical simulation of universal quantum computation (UQC) requires keeping track of vectors and matrices that describe the quantum system, whose dimensions grow exponentially in the system size. Methods scaling exponentially either in space or in time with the problem size are considered to be intractable. However, there are also restricted (i.e. non-universal) modes of quantum computation, such as Clifford circuits [11], Matchgates [12] or Tensor Networks [13] which are tractable for classical devices. These modes give us intuition as to which characteristics of quantum algorithms truly do (or do not) give them superior power over classical algorithms. Typically these restricted modes have a specific ingredient or resource that, when included in the computation, yield UQC again. An example is the addition of SWAP gates to Matchgate circuits [14].

Schuld *et al.* prove that UQC is necessary for a quantum model to be an universal function approximator, assuming exponential circuit depth [15]. With machine learning tasks it is of utmost important for the model to be able to represent the underlying system or function that generates the data distribution. Typically the underlying function is not known or fully understood, so universal function approximators are a safe choice to ensure successful learning. However this does not exclude the possibility non-universal but computationally cheaper quantum models are capable of representing the function, so we should ask ourselves if we really need fully quantum circuits in VQA, or whether some restricted circuit model is also sufficient for common machine learning tasks. To properly study the advantage gained by using fully universal quantum models in a learning setting one can compare models based on restricted modes of quantum computation with and without the additional resource that yields UQC. The methodological challenge is to find the most powerful classical simulation method for a restricted mode whereby one can control the amount of this resource in our model and exactly determine its computational hardness. A positive relation between the computational hardness and performance on learning tasks would provide solid justification for the use of quantum devices for machine learning.

In this work we probe the necessity of universal quantum computation in quantum machine learning, specifically VQAs. First we assess literature for various restricted modes of quantum computation and what additional resource or process they require to recover UQC. Based on these insights we create two original algorithms for the simulation of Matchgate circuits supplemented with SWAP gates. Our main result will be to show one of these represents the most powerful simulation technique reported in current literature. Additionally we develop a class of quantum models where we partition a large circuit into multiple smaller subcircuits. The exponential scaling of this class is now in the degree of non-local entanglement between the subcircuits, rather than the overall system size. All three methods provide a suitable basis to investigate how crucial it is to have fully quantum models in QML, however circuit partitioning approach is the only practically feasible method. We probe the necessity of UQC in VQA by exploring the relation between the simulation difficulty and learning performance. We develop a simulator for the partitioning approach and use it to test a common type of model seen in literature on various real-world and quantum datasets. We find our model already achieves competitive results in the absence of the non-local entanglement, and varying the simulation difficulty has no impact on the performance of the model.

The thesis is structured as follows. In Section 2 we introduce some theoretical preliminaries and definitions. In Section 3 we set out our research questions and original contributions. Related works are discussed in Section 4. Our new algorithms are presented and discussed in Section 5. Finally in Section 6 we present our experiments and results on the learning capabilities of our chosen class of models.

Chapter 2

Preliminaries

Before we go on to present our research questions it will be beneficial to introduce some concepts from Quantum Computing, Machine Learning and QML. Readers well versed in these topics may skip ahead to Section 3. Only concepts relevant to this thesis are discussed in a concise manner.

2.0.1 Quantum Computing

Whilst classical bits can be in either the 0 or 1 state, quantum bits exhibit a property called superposition whereby they can exist as a linear combination of the two basis states. In Dirac notation the quantum equivalent of the 0 basis state is $|0\rangle = \binom{1}{0}$ and the 1 basis state is $|1\rangle = \binom{0}{1}$. An arbitrary single qubit quantum state is described by a state vector, denoted as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix}, \tag{2.1}$$

with $\alpha, \beta \in \mathbb{C}$ and satisfy $|\alpha|^2 + |\beta|^2 = 1$. In fact, this state vector completely describes a closed physical system, and is a unit vector in the associated Hilbert space $\mathcal{H} = \mathbb{C}^2$. Larger systems comprising of n qubits are described by a state vector of size 2^n, reflecting the number of possible basis states.

Closed quantum systems (described by a quantum state) evolve according to the Schrödinger equation,

$$i\frac{d|\phi\rangle}{dt} = H|\phi\rangle, \tag{2.2}$$

where H is a hermitian operator, called the Hamiltonian, and we set $\hbar = 1$. A more practical but equivalent formulation is to say the quantum state evolves via unitary transformations,

$$|\phi'\rangle = U|\phi\rangle, \tag{2.3}$$

where $U = e^{-iHt}$ is a $2^n \times 2^n$ unitary matrix for a system of n qubits, and t is the time duration the system is being evolved by.

When combining N closed systems to obtain a composite system, one obtains the state vector by taking the tensor product of the state vectors of each individual system, $|\Phi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes ... \otimes |\phi_N\rangle$. More generally, the state space of a composite system is given by the tensor product of the individual state spaces $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes ... \otimes \mathcal{H}_N$. The opposite operation, to decompose a single system into multiple subsystems, is only possible if the subsystems are not *entangled*. A state is entangled if it cannot be written as a product of its component states, for example $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$.

Quantum measurements are the tool to obtain information from (closed) quantum systems. In particular here we focus on *projective measurements*, and refer the reader to [16] for a broader overview on quantum measurements in general. Projective measurements are described by an observable O, which is a Hermitian operator. Each eigenvector of O corresponds to a particular basis state of the system being measured, and has an associated eigenvalue λ. The spectral decomposition of $O = \sum_i \lambda_i P_i$ where P_i is the projection of the i^{th} eigenvector, which has a rank equal to the degeneracy of the eigenvalue. When measuring the state $|\phi\rangle$ the probability of obtaining the result λ_i is $p(\lambda_i) = \langle\phi|P_i|\phi\rangle$. If, after measurement, one obtains the outcome λ_i, the state collapses to the associated eigenvector:

$$\frac{P_i|\phi\rangle}{\sqrt{(p(\lambda_i)}}. \tag{2.4}$$

Often these observables O have some physical interpretation which teaches us something about the system if we calculate the expected value $\langle O\rangle = \langle\phi|O|\phi\rangle$ of the observable. Due to the collapse of the wavefunction, to obtain the expected value experimentally one has to initialize the system, evolve it, and measure it repeatedly. After many measurements one obtains an estimate of the expectation value of the observable by taking an average over all the observed measurement outcomes.

Deutsch introduced the circuit model of quantum computation, whereby the evolution of a quantum state is governed by the sequential action of a set of unitary gates [17, 18]. A universal gate set is a set of fixed and/or parameterized gates that is capable of approximating any unitary evolution to arbitrary precision. The set of Pauli matrices, which we shall denote I, X, Y and Z form a set of very elementary gates. More common single-qubit gates include the Hadamard gate H, and the parameterized Pauli rotations $R_P(\theta) = \exp(-iP\theta/2)$ where P is one of the Pauli matrices,

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad R_X(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}.$$

Common two-qubit gates are the CNOT and SWAP gate, with the following matrix representations;

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Quantum circuits consist of wires that carry qubits, and quantum gates that are executed in a particular order. An example of a circuit diagram is given in Fig. 2.1.

As stated earlier, a quantum state consisting of n qubits is fully described by a vector of dimension 2^n, whilst its evolution is governed by unitary matrices of size $2^n \times 2^n$. As a result it is possible to carry out "brute force" simulation of such a system on classical computers using Schrödinger's algorithm [19]. The exponential spacial complexity makes it is near impossible to study systems consisting of tens of qubits or more. Feynman's algorithm offers an alternative which is linear in memory,

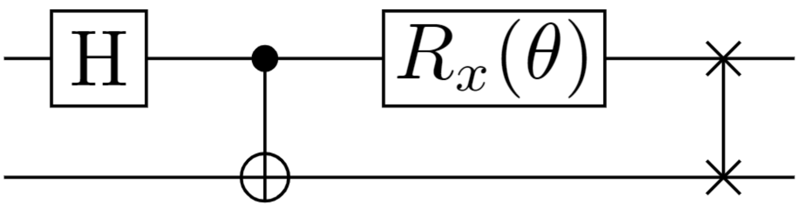


Figure 2.1: A diagram of a two-qubit circuit containing a Hadamard gate, a CNOT, a Pauli-X rotation and a SWAP gate respectively. It is for illustrative purposes only, and does not represent an algorithmically meaningful sequence of gates.

but exponential in the time complexity. An exception here are systems which can be decomposed into multiple subsystems. Simulation techniques that make use of this feature are often referred to as "Divide & Conquer" methods. Restricted modes of quantum computation (such as those mentioned in the introduction) are classes of quantum systems whereby the classical resources required to simulate them only grows polynomially in the system size, denoted poly(n). These classes can be supplemented with a particular resource (or in the circuit model of quantum computation, a specific gate) to recover UQC. An example of such a system are non-interacting fermions. These are defined by the formalism of Matchgates, which describes the action of two-qubit parity-preserving unitary operators allowed to act on nearest neighbour qubits only. The addition of SWAP gates is required to obtain universal quantum computation again. We will explore this formalism in Section 5.1. Since classical simulation of UQC has exponential scaling in the input size, we can expect the classical simulation techniques of the SWAP gate within the matchgate formalism to also generate exponential scaling, now in the quantity of SWAP gates rather than the overall system size.

It is important to consider what a simulation algorithm is capable of outputting. In this report we will distinguish between the *strong*, *semi-strong* and *weak* classes of classical simulation. These definitions are based on the notions of strong and weak classical simulation from [20]:

Definition 2.0.1 (Strong simulation)**.** Consider a uniform family of quantum circuits $\{C_n\}$ acting on an n qubit input state $|\Psi\rangle$. $\{C_n\}$ is classically simulable in the strong sense if, for every possible final measurement outcome y of k qubits in the computational basis, the distribution $\Pr(y|\Psi)$ can be computed to a precision of m digits on a classical computational device.

Brute-force simulation belongs this class, as it keeps track of the full state vector throughout the computation. As this vector completely describes the closed system, it is possible to extract any and all information desired, if necessary by creating copies of the vector.

Definition 2.0.2 (Semi-strong simulation)**.** Consider a uniform family of quantum circuits $\{C_n\}$ acting on an n qubit input state $|\Psi\rangle$. $\{C_n\}$ is classically simulable in the semi-strong sense if, for some observable O on k qubits, the expectation value $\langle O\rangle$ can be computed to a precision of m digits on a classical computational device.

Definition 2.0.3 (Weak simulation)**.** Consider a uniform family of quantum circuits $\{C_n\}$ acting on an n qubit input state $|\Psi\rangle$. $\{C_n\}$ is classically simulable in the weak sense if, for every possible final measurement outcome y of k qubits in the computational basis, the distribution $\Pr(y|\Psi)$ can be sampled from on a classical computational device.

Whilst strong simulation is a more powerful tool, it is actually weak simulation which more closely reflects the use of a quantum device. Measurement of a quantum device is only capable of returning a sample from the probability distribution defined by the state vector as a consequence of Eq (2.4). It is also not possible to repeatedly sample copies of the state vector since copying quantum states is forbidden by the No-cloning theorem. We consider the simulation classically efficient if it can be done in polynomial time and space in n, k and m on a classical device.

2.0.2 Machine Learning

The goal of Machine Learning (ML) is to develop algorithms that can learn to perform some task from data, and consequently come to some conclusions autonomously. There are three main paradigms in ML:
Supervised learning, where algorithms learn a function which maps some input to an output using labelled examples. The can be of images, text, real-world observations or similar;
Unsupervised learning, where the dataset is does not contain labels, but rather the task is to find structures or patterns in the given examples, typically with the goal of finding a compressed representation of the data;
Reinforcement learning, where algorithms learn by repeated interaction with an environment in a stochastic process. Feedback is given in the form of rewards, and the goal is to maximize the reward accumulated over throughout the process.
As we are using supervised learning to test our algorithms in this report, let us discuss this paradigm in more detail.

The goal of supervised learning is to make predictions on unseen data (a test set), having learnt from a set of given examples (the training set). Both sets contain feature vectors and their corresponding label. This data is drawn from some random variable X with underlying probability distribution P_X, which is a mathematical model of the underlying observation procedure. The sample space S_X denotes the set of all samples that can be drawn from X. The prototypical task in supervised learning is to learn the decision function $h : S_X \to S_Y$, based only on some model ansatz $\tilde{h}$ and a finite set of data points $(\mathbf{x}_i, y_i)_{i=1,..,N}$, where $\mathbf{x}_i \in S_X$ is the feature vector and $y_i \in S_Y$ the label. The common approach is to construct a parameterized model, or more formally a hypothesis class:

Definition 2.0.4 (Hypothesis class)**.** A hypothesis class is a family of surjective decision functions defined by some model ansatz $\tilde{h}$ parameterized by $\boldsymbol{\theta}$ which map samples from variable X onto the sample space of Y:

$$\mathbb{H} = \{\tilde{h}(\boldsymbol{\theta}) : S_X \to S_Y | \boldsymbol{\theta} \in \mathbb{R}^M\}$$

where M is the number of parameters in the model.

In order to quantify the quality of the model ansatz one uses a loss function $L : S_Y \times S_Y \to \mathbb{R}^{\geq 0}$ which returns some value based on the closeness of the output of the function $\tilde{h}(\boldsymbol{\theta}, \mathbf{x}_i)$ to the true label y_i. Learning the optimal decision function now comes down to minimizing the *true risk*,

$$R^{true}(\tilde{h}) = \mathbb{E}[L(\tilde{h}(x), y)] \quad \forall (x, y) \sim X \times Y. \tag{2.5}$$

As we do not have access to the distributions underlying X and Y, but rather a set of N data samples, we instead deal with the *empirical risk*,

$$R^{emp}(\tilde{h}) = \frac{1}{N}\sum_{i}^{N} L(\tilde{h}(\mathbf{x}_i), y_i). \tag{2.6}$$

This corresponds to computing the average loss for a set of predictions and true labels. The function $\tilde{h}^*$ which minimizes this empirical risk can be found using optimization algorithms,

$$\tilde{h}^*_{\boldsymbol{\theta}} = \underset{\tilde{h}\in\mathbb{H}}{\text{argmin}} \frac{1}{N}\sum_{i}^{N} L(\tilde{h}(\mathbf{x}_i), y_i). \tag{2.7}$$

Since each decision function $\tilde{h}$ is defined by its parameters $\boldsymbol{\theta}$ we have $\tilde{h}^* = \tilde{h}(\boldsymbol{\theta}^*)$, allowing us to re-write the above equation as

$$\tilde{h}^*_{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}^*\in\mathbb{R}^{\dim(\boldsymbol{\theta})}}{\text{argmin}} \frac{1}{N}\sum_{i}^{N} L(\tilde{h}(\mathbf{x}_i), y_i). \tag{2.8}$$

As the primary objective of supervised learning is to minimize the true risk, we must recognize that minimizing the empirical risk does not guarantee we complete this objective. We gauge the difference between the true and empirical risks through a measure called the generalization loss (sometimes referred to as the generalization performance). The generalization loss is defined as follows

$$G_L = R^{Emp}_{test} - R^{Emp}_{train}, \tag{2.9}$$

the difference between the loss on the training (R^{Emp}_{train}) and test R^{Emp}_{test} sets. If the generalization loss is small, we assume we have successfully minimized the true risk. On the other hand, if there is a large discrepancy the model has likely over-fit to the training data, and has failed to learn the true decision function h. For convenience we sometimes use the *generalization error*, which is the model accuracy on the training set minus the accuracy on the test set.

The choice of loss function can vary based on the type of label under consideration. In the context of classification, each data point belongs to one particular class c_i from a discrete set of classes C. The output of a decision function in this setting can be interpreted as the probability that that data point belongs to a particular class (given the output of the decision function is in the range $(0, 1)$). A common choice of loss function for classification task is Cross Entropy. When there are only two unique classes, the Binary Cross Entropy Loss is given as

$$L_{\text{BCE}}(\mathbf{x}_i, y_i) = -\big(y_i\log(p(y_i|\mathbf{x}_i)) + (1-y_i)\log(1-p(y_i|\mathbf{x}_i))\big). \tag{2.10}$$

For regression tasks using continuous, real-valued labels, loss functions like the Mean Square Error can be more appropriate:

$$L_{\text{MSE}}(\mathbf{x}_i, y_i) = (\tilde{h}(\mathbf{x}_i) - y_i)^2. \tag{2.11}$$

2.0.3 Quantum Machine Learning

Variational quantum circuits refers to the notion of quantum circuits containing adjustable gates (such as the Pauli rotation gates), and possibly also fixed gates (eg the Hadamard or CNOT gates). We can denote the action of such a circuit as $U(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the collection of parameters defining the adjustable gates. Such circuits are suitable model candidates for a hypothesis class [21]. To complete the hypothesis class we need a data encoding strategy, which for now can be absorbed into the circuit itself $U(\boldsymbol{\theta}, \boldsymbol{x})$, and some procedure to obtain a (classical) label from the state returned by the circuit. A logical choice is to use the expectation value of some observable O, giving the following hypothesis class $\{\tilde{h}(\boldsymbol{\theta}, \boldsymbol{x}) = \langle\phi|U^{\dagger}(\boldsymbol{\theta}, \boldsymbol{x})OU(\boldsymbol{\theta}, \boldsymbol{x})|\phi\rangle\}$ for some arbitrary fiducial state $|\phi\rangle$. For classification tasks an additional classical function $g : \text{domain}(\tilde{h}) \to \{c_i\}$, might be necessary to return a label from the set of class labels $\{c_i\}$.

Various VQC based quantum models have been proposed in literature. In 2018 M. Schuld *et al.* and Mitarai *et al.* both developed the idea of using a shallow VQC for supervised learning [22, 23], and demonstrate learning for a variety of artificial and real-world datasets. Havlicek *et al.* were the first to experimentally realize this idea on a quantum computer [24]. The feasibility of this approach is supported by the existence of an analytical expression for the gradients of single qubit parameterized gates. The expression is referred to as the parameter shift rule, and is especially useful for NISQ devices where the errors of numerical differentiation are compounded by noise. Deploying such a quantum ansatz for supervised learning leads to a quantum-classical hybrid system where the role of the quantum computer is to output labels (and gradients based the input data and parameters for variational models). The overall learning process takes place in a hybrid quantum-classical system where the role of the quantum processing unit (QPU) is to perform state preparation and measurement [21]. By repeating the state preparation and measurement multiple times an estimate of the expectation value of some observable can be made, which can be postprocessed to generate a label. The classical central processing unit (CPU) feeds in data and model parameters governing the state preparation (and possibly also the measurement) to the QPU, and also performs the optimization loop to minimize or maximize an objective function. The interplay of the two devices is the backbone of a class of algorithms called Variational Quantum Algorithms (VQA), which extend far beyond to domain of Quantum Machine Learning [5].

Chapter 3

Research Questions & Original contributions

In this thesis we aim to investigate how the classical computational hardness of a variational quantum circuit model influences its machine learning properties. We do this using a Variational Quantum Algorithm, where the model parameters are optimized using a classical procedure. To quantify and control the computational hardness we must understand the fundamental quantum aspects of quantum models; what features inhibit their efficient classical simulation? The more intuitive approach is to ask what additional features are required by classically simulable restricted quantum systems such that they yield universal quantum computation. Based on literature discussing this matter, we ask ourselves:

1. *Can we develop simulation methods whereby we systematically control the "quantumness" i.e. the abundance of the resource required for universality, of a quantum system?*

Whilst such simulation methods exist in literature, we find there is room to develop new methods. The challenge is developing these methods within the framework of the formalism of the corresponding restricted mode of QC. This is necessary to maintain its efficient classical simulation in the absence of any additional resources. Typically the abundance of the relevant resource is a countable quantity, such as the number of times a particular gate appears in a circuit e.g. SWAP gates in Matchgate simulations. In order to carry out numerical studies using these simulation methods we probe the following question:

2. *Can we construct a sufficiently efficient classical simulator of these methods such that we can simulate meaningful quantities of the additional resource that yields UQC?*

After tackling this question we can systematically control the classical computational hardness of a variational quantum model, allowing us to address the final research question:

3. *Does the computational hardness of a family of quantum models influence its performance in a machine learning setting?*

In answering these questions we deliver the following original contributions. We:

1. Propose a novel method of simulating SWAP gates in the Matchgate formalism whereby one decomposes the SWAP gate into a sum of Matchgates. Building on established Matchgate simulation techniques, our method gives rise to two new algorithms, one of which is of a stronger level of simulation than the gadget-approach for the same computational scaling, and also benefits from a favourable overhead.

2. Introduce a class of quantum models based on a Divide and Conquer scheme introduced by Bravyi [25], which gives a scaling that is linear in the total number of qubits and exponential in the non-local entanglement. We dub this class of models "Partitioned models".

3. Develop a comprehensive classical simulator of the class of Partitioned models and a pipeline to test their learning performance. The code can be found on Github [26].

4. Demonstrate the learning performance of the efficiently simulable class of Partitioned models is competitive with classical methods, and not simply related to the computational hardness of the model.

We assess the learning performance on a variety of real and artificial datasets.

Chapter 4

Related Work

The work of Bravyi *et al.* develops three classical algorithms to calculate the quantum mean value of a circuit [27]. The quantum mean value problem refers to the calculation of the expectation value of a n-qubit tensor product observable on the output of a shallow quantum circuit. This task is a cornerstone in many QML algorithms. Each algorithm introduces a necessary restriction to achieve a computational speedup. This restriction can significantly impede practical applications of the algorithm, but the implications are insightful nonetheless.

The simplest algorithm applies to observables O_j close to the identity, $||O_j - I|| \leq \mathcal{O}(2^{-5d})$, where d denotes the circuit depth. Observables satisfying this restriction could be used for verification purposes of noisy devices. The restriction based on the circuit depth implies the method is only efficient for circuits which at most grow logarithmically with the number of qubits. The second algorithm works only for positive semi-definite observables and constant depth circuits. Interestingly, this algorithm can output the absolute value of some expectation value of a Hermitian (not necessarily positive semi-definite) observable, which leaves just the sign of the mean value to be determined. The final algorithm works for an arbitrary observable but the circuit must be defined on a two or three dimensional grid of qubits with a constant depth circuit containing at most nearest-neighbour gates. Such mean value problems can be solved in time $\mathcal{O}(n)$ and $2^{\mathcal{O}(n^{1/3})}$ for 2D and 3D grids respectively, however the method suffers from a large constant overhead preventing it from being a useful simulation method.

These results are suggestive of the features of quantum algorithms that lead to quantum (dis-)advantages over classical methods. Either it must have super-constant circuit depth (e.g. logarithmic in the number of qubits is already sufficient), require qubit connectivity graphs that do not permit embedding in a 2D grid or entanglement increasing with the system size.

In a landmark paper in 2002, Valiant introduced matchcircuits as a restricted mode of quantum computation that can be simulated classically in polynomial time [28]. These early "matchcircuits" were quantum circuits generated by two-qubit operators where the 16 matrix entries were governed by a set of five polynomial equations. Terhal & DiVincenzo showed the equivalence between these matchcircuits and nearest-neighbour unitary linear fermionic operators. Furthermore, they showed that the addition of arbitrary single qubit gates to the gate set gives rise to a universal mode of quantum computation [12]. Jozsa & Miyake consequently identified that relaxing the nearest-neighbour condition to next-nearest-neighbour also gives universal computation, meaning qubit swapping (i.e. the SWAP gate) is in some sense the boundary between classical and quantum computational power [14]. Brod & Galvão extended this result (and boundary) to show any parity-preserving non-matchgate unitary is capable of promoting matchgates to universal quantum computation [29]. The work of Hebenstreit *et al.* provides a broad overview of the various classical simulation methods of Matchgates, and possible additional features (e.g. product state inputs or adaptive mid-circuit measurements) which when in limited supply remain efficiently simulable [30]. We conclude with the remark that it has been proven that an n qubit matchgate circuit can be compressed and run on a universal quantum computer using just $\mathcal{O}(\log(n))$ qubits [31].

There has been some research done into hybrid systems whereby a larger quantum circuit is simulated using a small quantum computer aided by some classical resources. Such schemes use Divide and conquer (D&C) or circuit cutting methods, and are an important line of research to maximize the utility of currently available NISQ devices [32]. These methods typically introduce some sort of cut that allows the whole system to be treated as a weighted linear sum of smaller composite systems. Generally, these algorithms scale as $2^{\mathcal{O}(K)}\text{poly}(n)$ for some parameter K related to the number of cuts, and n qubits [25]. The benefit of such methods mainly arises when the whole system is too large to be handled by the available resources, but the subsystems are not. There is also some evidence that circuit cutting methods lead to higher fidelity estimates of the output of a circuit compared to full circuit execution [33], which can be understood from the reduced noise due to the smaller circuit size.

Various forms of cut have been investigated:
Tensor networks Tensor networks intuitively lend themselves well to partitioning methods [34]. Both Yuan *et al.* and Barratt *et al.* use hybrid D&C methods to solve problems where tensor networks make for a very efficient representation of the target system, allowing them to solve large scale systems with only a small quantum computer [35, 36]. Peng *et al.* focus on the simulation of quantum circuits. By first finding the tensor network representation of the circuit and cutting some edges of the network, the resulting smaller networks can be converted back into quantum circuits, now acting on fewer qubits [37]. Clearly the scaling parameter K is circuit dependent, and the authors recognize their method lends itself well to ansatze with a high degree of clustering. They are able to implement a 6-qubit VQE on a 3-qubit device.

Hamiltonian reduction Fujii *et al.* propose a two-level VQE method for Hamiltonians containing strong interactions within different subsystems of qubits, but weak interactions between these subsystems [38]. By first solving each subsystem in the absence of intersubsystem action, an approximate ground state can be built up. In the second step, this approximate state is used to generate a basis with fewer degrees of freedom to construct an effective Hamiltonian which does consider the intersubsystem terms, which is consequently solved again using VQE (but now its called *deep VQE*).

Qubit wires Tang *et al.* develop a full end-to-end method where a qubit wire is cut vertically at some point in the circuit, creating smaller subcircuits which are accessible to- and executed on available quantum devices [32]. Classical postprocessing is then used to reconstruct the output of the original circuit. An example cutting of a circuit is shown in Fig. 4.1. The authors complement their framework

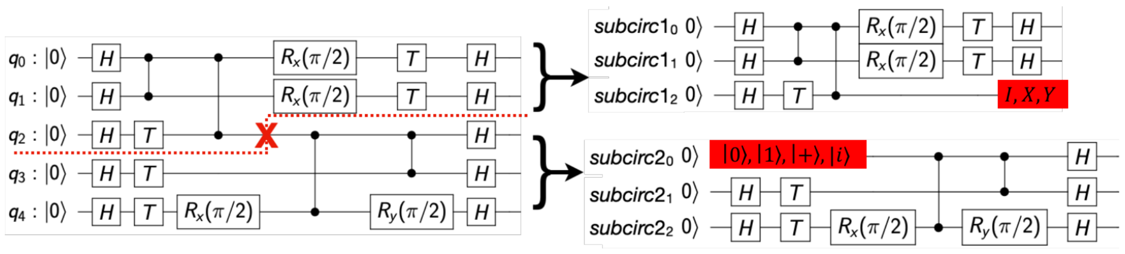


Figure 4.1: Schematic of the qubit wire cutting procedure of Tang *et al.* The original five-qubit circuit in the left hand diagram is cut at the point marked by the red cross. The resulting subcircuits on the right hand side are evaluated for the different operators or states shown in the red box. The final output can be reconstructed using classical postprocessing. Figure reproduced from [32].

with a method to find the optimal cuts for arbitrary input quantum circuits.

Gate decomposition Rather than making a vertical cut as in the work of Tang discussed above, it is also possible to cut circuits in the horizontal direction. Specifically this involves "cutting" two-qubit gates in half, whereby you decompose the gate into a sum of tensor products of unitaries acting on each qubit individually [25]. In the Pauli basis this can give up to 16 terms, so by linearity of the inner product on has to calculate $\mathcal{O}(16^{2K})$ inner products for K two-qubit gates cut, although it is often the case the decomposition requires fewer terms. As an example consider the left hand circuit in Fig. 4.1. By cutting the cZ gate acting on qubits q_0 and q_2 one is able to partition the circuit into two blocks, q_0 and q_1, and q_2, q_3 and q_4, as there are no gates acting across the boundary of these blocks. This technique generally requires an unreasonable number of circuit evaluations for any substantial number of cuts. Marshall *et al.* counteract this by accepting to merely approximate the output of the original circuit with a limited budget of circuit evaluations [39]. They do this by first parameterizing the decomposed two-qubit gates such that different values of the parameter correspond to specific terms in the decomposition. This mitigates the exponential number of inner products to be estimated at the cost of increasing the number of parameters, which in the context of Machine Learning is an acceptable compromise where gradient based optimization can effectively deal with large parameter spaces.

Whilst all the above works focus on utilizing quantum computers, their methods can readily be applied in the context of classical simulation too. As the cost of strong simulation of quantum circuits increases exponentially with the system size, these D&C techniques can readily increase the overall system size that can be handled on classical devices.

In the field of QML there is an active debate on what grounds we should compare quantum to classical methods. Where early on the focus was on improvements in the algorithmic scaling [40], current research places more emphasis on the near term feasibility of the model. This has led to heuristic approaches such as the VQA method considered in this work. It is still possible to prove quantum advantage using these heuristic approaches. Liu *et al.* constructed a classification problem based on the Discrete Logarithm Problem for which Shor's algorithm offers an exponential speedup over classical methods [41]. Using a quantum device to compute the kernel for a Support Vector Machine (SVM) they are able to realize this quantum advantage. This kind of artificial construction has a very limited applicability to real-world problems, which motivates analysis based on the capacity of the quantum model [10].

Analysis of the capacity of quantum models in literature uses established techniques from classical learning theory. Schuld proved quantum models based on some encoding circuit (i.e. a circuit whose parameters are a function of the data) followed by a parameterized section $U(\boldsymbol{\theta})U(\mathbf{x})|\mathbf{0}\rangle$ are a special case of the well-understood kernel methods [42]. This relation gives a very natural environment to discuss quantum advantage, as one can devise quantum "feature maps" that are classically hard to simulate based on their output data distributions [24]. Jerbi *et al.* develop a framework which demonstrates that implicit quantum models (which use $|\langle\mathbf{0}|U^\dagger(x)U(x')|\mathbf{0}\rangle|^2$ directly for labelling) can achieve zero empirical risk, but fail to minimize the true risk. Within the same framework explicit models (those where labels are assigned based on the output of $U(\boldsymbol{\theta})U(\mathbf{x})|\mathbf{0}\rangle$) introduce the necessary restriction on the expressive power to yield a nontrivial generalization error. More general still are data re-uploading methods, where the data is encoded multiple times, either into additional qubits, or additional layers. Theoretical analysis by Schuld *et al.* show data re-uploading enriches the frequency spectrum and enlarges the range of the coefficients of the Fourier representation of the circuit [15]. Other measures of capacity proposed in literature include the Fisher information spectrum [43], Rademacher complexity [44] and Vapnik–Chervonenkis dimension [45]. The latter two can be used to establish theoretical bounds on the generalization performance of the model. This is a useful tool to design models with a high capacity, although again this does not guarantee the model will perform well in practical applications.

Beyond small-scale benchmarks, real world datasets often have a large number of features. There has only been a modest amount of research into the performance of quantum models on high-dimensional data. Peters *et al.* use a quantum SVM (QSVM), which are a kind of implicit model, to learn 67-dimensional data on up to 17 qubits [46]. Their noiseless simulations demonstrate overfitting, whilst for results from their hardware the training and validation error lie much closer together. Haug *et al.* achieve a similar feat, but instead use randomized measurements (and postprocessing) to quadratically reduce the number of measurements required to compute the kernel [47]. Both papers encode features in single qubit rotations, and subsequently encode multiple features per qubit.

Chapter 5

Classical Simulation of quantum systems

Whilst the development of quantum devices is currently making steady progress, simulating them using classical computers remains an important tool to study quantum algorithms. One goal of this work is to find a family of quantum models where we have full control over the computational hardness of the simulation of the model. Let us be more specific with what we mean by simulating a model, for the purposes of this work. We want a general circuit simulator which takes as input:

1. some well-defined quantum circuit U on n qubits,
2. a set of parameters $\boldsymbol{\theta}$ which dictate the action of certain gates in the circuit,
3. an observable f.

The output of this algorithm should be either an estimate or the exact value of the expectation value of the observable on the output of this circuit $\langle \mathbf{0}|U^\dagger(\boldsymbol{\theta})fU(\boldsymbol{\theta})|\mathbf{0}\rangle$, corresponding to weak and (semi-)strong simulation (definitions 2.0.3, 2.0.2 & 2.0.1) respectively. This can be done by computing sequentially the action of each gate in $U(\boldsymbol{\theta})$ on the $|\mathbf{0}\rangle$ fiducial state, yielding some output state $|\Psi_{\text{out}}\rangle$. The expectation value can then be computed explicitly $\langle\Psi_{\text{out}}|f|\Psi_{\text{out}}\rangle$. Such an approach leads to a complexity scaling exponentially in the number of qubits. We would like a circuit simulator with either the space- or time complexity to scale exponentially in the number of resourceful gates and polynomially in the number of qubits and other gates.

The first candidate for our circuit simulator are Matchgate circuits, which have a rich theory regarding its computational power associated to it. Pure Matchgate computations are classically efficiently simulable, and there are multiple ways to extend the theory to achieve full UQC [30], such as the addition of SWAP gates (a resourceful gate in the Matchgate formalism). One can exactly choose the number of SWAP gates in a variational Matchgate model, offering us a way to systematically increment the computational hardness of the model and satisfy research question 1. Furthermore, Matchgates are naturally parameterized to generate a continuous family of functions which makes them particularly well suited for QML and to help us answer research question 3. It remains to find a simulation method of Matchgate circuits containing SWAP gates which allows us to experiment with circuits containing more than two or three SWAP gates, as demanded by research question 2.

Let us provide an overview of some Matchgate simulation methods from literature. For pure Matchgate circuits the main simulation methods are the Heisenberg technique and Wick's simulation. At the core of both methods are Majorana operators, which are related to the fermionic annihilation and creation operators as $c_{2i} = a_i + a_i^\dagger$ and $c_{2i+1} = -i(a_i - a_i^\dagger)$ where i indexes the qubit. Simulations of Matchgates are restricted to observables that are products of Majorana operators (or fermionic annihilation and creation operators). The number of factors in the product affects the simulation cost which necessitates introducing the notion of the *rank of an observable*, D (not to be confused with the rank of a matrix). The rank of an observable is the number of factors in the product that defines the observable. For example a single Pauli-Z observable on the i-th qubit is proportional to $c_{2i-1}c_{2i}$, which is of rank $D = 2$. We discuss the Heisenberg technique and Wick's simulation in more detail in Section 5.1.2 and 5.1.3 respectively as they form an important basis for the simulation methods which also allow the addition of the SWAP gate in the circuit.

To simulate Matchgate circuits containing also a few SWAP gates Hebenstreit *et al.* introduced the SWAP gadget [48]. The gadget allows one to implement the SWAP gate within the Matchgate formalism by using a combination of Matchgates, so-called "magic states" and adaptive mid-circuit measurements. We defer the details and proof to Section 5.1.4 and simply state the time complexity of the algorithm:

Theorem 5.0.1 (SWAP Gadget, [48])**.** *A quantum circuit on n qubits consisting of polynomially many Matchgates and N SWAP gates, paired with some observable of rank D can be weakly simulated using the SWAP gadget with the time complexity upper bounded by $\mathcal{O}(4^{2N}(D + 24N)^3 \mathrm{poly}(n))$.*

This method represents the state-of-the-art simulation method of Matchgate circuits containing SWAP gates in literature. Importantly, it is only able to provide an estimate of the expectation value as the the final simulation output must be evaluated for the various possible mid-circuit measurement outcomes. This corresponds to the weak level of simulation, meaning one would have to repeat the simulation many times to obtain a reasonable estimate of the expectation value.

Having to repeat the simulation multiple times to obtain the output hinders the practical usage of the method. An important contribution of our work to overcome this is the development of two new methods to implement SWAP gates in Matchgate computations which correspond to the semi-strong level of simulation. Our approach is to decompose the SWAP gate into a sum of Matchgates, which allows one to represent a Match- and SWAP gate circuit as a sum of Matchgate-only circuits. The desired expectation value can be computed by evaluating the inner products between the various Matchgate-only circuit as this is a linear computation. We prove these inner products can be evaluated using both the Heisenberg technique and Wick's simulation in the following two theorems.

Theorem 5.0.2 (Divide & Heisenberg)**.** *A quantum circuit on n qubits consisting of polynomially many Matchgates and N SWAP gates, paired with some observable of rank D can be semi-strongly simulated using the Divide & Heisenberg method with the time complexity upper bounded by $\mathcal{O}(4^{2N} n^{D+4N} \mathrm{poly}(n))$.*

Theorem 5.0.3 (Divide & Wick). *A quantum circuit on n qubits consisting of polynomially many Matchgates and N SWAP gates, paired with some observable of rank D can be semi-strongly simulated using the Divide & Wick method with a time complexity upper bounded by $\mathcal{O}(4^{2N}(D + 4N)^3 \text{poly}(n))$.*

The proof of Theorems 5.0.2 and 5.0.3 are provided in Section 5.1.6 and 5.1.7 respectively. As these theorems rely on the Heisenberg technique and Wick's simulation we dub the associated simulation technique "Divide & Heisenberg" and "Divide & Wick".

Algorithm	**Time complexity**	**Simulation type**
SWAP gadget + Wick's Simulation	$4^{2N}(D + 24N)^3\text{poly}(n)$	Weak
Divide & Heisenberg†	$4^{2N}n^{D+4N}\text{poly}(n)$	Semi-strong
Divide & Wick†	$4^{2N}(D + 4N)^3\text{poly}(n)$	Semi-strong

Table 5.1: A comparison of three algorithms for the simulation of Matchgate circuits containing SWAP gates. The time complexity is parameterized by the number of SWAP gates, N, the rank of the observable D, and the total number of qubits in the circuit n. Each algorithm is discussed in Section 5.1.4, 5.1.6 and 5.1.7 respectively. † identifies our contributions.

We summarize the main differences between the three Match- and SWAP gate circuit simulation techniques in Table 5.1. From the time complexity it is evident methods based on Wick's theorem offer the best scaling. The scaling between the SWAP gadget and Divide & Wick are very similar, however the latter is of a stronger level of simulation. Consequently, in tasks where one wants to compute the expectation value of some observable our Divide & Wick method is superior over the SWAP gadget. Despite the Divide & Wick being the most powerful Match- and SWAP gate circuit simulation method available, the exponential scaling in the number of SWAP gates is still prohibitively expensive to practically use a Matchgate circuit as a quantum model. Although it is difficult to quantify the exact number of SWAP gates one could simulate without empirical testing, it is likely too few to satisfy research question 2.

Our second candidate circuit simulator is the Circuit Partitioning formalism. It is inspired by the gate decomposition techniques of [25] and [39]. Consider a variational quantum circuit containing primarily single-qubit gates, and some CNOT gates. In the Circuit Partitioning formalism we partition the circuit into sets of qubits of equal size. CNOTs which act on qubits in different sets (i.e. across a partition) are then the resourceful gate. They are decomposed using the Schmidt decomposition, allowing us to simulate each circuit on subsets of qubits individually. Based on the Circuit Partitioning formalism we arrive at the following theorem:

Theorem 5.0.4. *The space complexity of a quantum circuit consisting of B partitions of n_p qubits and N CNOT gates acting across partitions is $\mathcal{O}(2^{n_p+N}B)$.*

Proof of this theorem and details of the Circuit Partitioning formalism can be found in Section 5.2. This method scales exponentially in the number of resourceful

CNOT gates, and linearly in the number of qubits when the partition size is fixed. In the absence of CNOTs we have a classically efficiently simulable model, and can increment the computational hardness by adding a single cross-partition CNOT. Secondly, the scaling gives a high degree of flexibility to balance the number of partitions with the number of cross-partition CNOT gates, which allows us to probe a satisfactory quantity of resourceful CNOT gates. Finally, the single-qubit gates can be parameterized to generate a continuous family of functions apt for a hypothesis class. Overall, the Circuit Partitioning formalism gives us a positive response to research questions 1 & 2 and offers a suitable platform to probe research question 3.

In the remainder of this chapter the details of the Matchgate algorithms are discussed in Section 5.1. Section 5.2 presents the Circuit Partitioning formalism.

5.1 Matchgates

We have identified Matchgate circuits as a suitable quantum model to investigate the research questions of this work. Research question 1 is focused around finding techniques whereby you supplement matchgate simulations with SWAP gates, while research question 2 demands minimizing the computational cost of doing so. In this section we start by introducing common notation used in the Matchgate setting. Following this we explain the Heisenberg technique and Wick's simulation, two core Matchgate simulation algorithms in Subsections 5.1.2 and 5.1.3 respectively. Subsequently we illustrate the workings of the SWAP gadget and prove Theorem 5.0.1 in Subsection 5.1.4. The basic premise of our SWAP gate decomposition is defined in Section 5.1.5, and prove Theorem 5.0.2 (Divide & Heisenberg) and 5.0.3 (Divide & Wick) in Subsections 5.1.6 and 5.1.7. We compare and discuss the simulation methods based on the three theorems in Subsection 5.1.8.

5.1.1 Notation

A 2-qubit matchgate is defined as

$$G(A,B) = \begin{pmatrix} a & 0 & 0 & b \\ 0 & e & f & 0 \\ 0 & g & h & 0 \\ c & 0 & 0 & d \end{pmatrix} \text{ with } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, B = \begin{pmatrix} e & f \\ g & h \end{pmatrix}, \tag{5.1}$$

where both A and B are in U(2) and det(A)=det(B). For Matchgate circuits to be efficiently simulable the qubits must be on a line and the gates may act only on nearest neighbour qubits.

A matchgate circuit is described by a unitary with a Hamiltonian quadratic in these Majorana operators [14]:

$$H = i \sum_{\mu \neq \nu = 1}^{2n} h_{\mu\nu} c_\mu c_\nu. \tag{5.2}$$

$h_{\mu\nu}$ is a $2n \times 2n$ matrix of coefficients. c_μ and c_ν are Majorana operators: for n qubit lines we have the set of $2n$ hermitian Majorana operators $\{c_\mu\}$. These are based on the fermionic creation and annihilation operators:

$$c_{2i} = a_i + a_i^\dagger, \qquad c_{2i+1} = -i(a_i - a_i^\dagger) \tag{5.3}$$

which satisfy

$$\{c_\mu, c_\nu\} = 2\delta_{\mu\nu} I, \tag{5.4}$$

where δ is the Knonecker delta. The Jordan-Wigner representation of the operators is as follows:

$$\begin{aligned} c_{2i-1} &\Leftrightarrow Z^{\otimes i-1} \otimes X \otimes I^{\otimes n-i} \\ c_{2i} &\Leftrightarrow Z^{\otimes i-1} \otimes Y \otimes I^{\otimes n-i}. \end{aligned} \tag{5.5}$$

Such a Hamiltonian is called a *quadratic Hamiltonian*, and $U = \exp(iH)$ a *Gaussian* operation. By the anti-commutation relations between the c_μ's, and requiring the Hamiltonian to be hermitian, we see $h_{\mu\nu}$ is real and antisymmetric.

The aim of Matchgate simulations is to compute the quantity $\langle \mathbf{0}|U^\dagger f U|\mathbf{0}\rangle$, where U is some Gaussian operation, and f an observable that can be expressed as a product of Majorana or fermionic operators (or a sum of products). This covers many common observables. A Pauli-Z on the ith qubit can be written as $-ic_{2i-1}c_{2i}$, and the single qubit projectors $|0\rangle\langle 0|_i = a_i a_i^\dagger$ and $|1\rangle\langle 1|_i = a_i^\dagger a_i$. All three expressions are a product of two Majorana or fermionic operators, and hence the *rank* of these observables is $D = 2$. More complicated observables can be constructed using these expressions. Note, based on these expressions generally we have D is twice the number of qubits being measured.

We are now ready to consider the Heisenberg simulation technique.

5.1.2 Heisenberg technique

The simplest approach to simulating a matchgate-only circuit is using the *Heisenberg technique*, which uses the following theorem by Jozsa & Miyake:

Theorem 5.1.1 ([14]). *Let H be any quadratic Hamiltonian and $U = e^{iH}$ the corresponding Gaussian operation. Then for all μ:*

$$U^\dagger c_\mu U = \sum_{\nu=1}^{2n} R_{\mu\nu} c_\nu,$$

where the matrix R is in SO(2n), and we obtain all of SO(2n) in this way. In fact $R = e^{4h}$, where h is the matrix of coefficients defined by the Hamiltonian H as per Eq. (5.2).

For the proof we refer the reader to [14]. Note that U (or it's associated Hamiltonian H) can represent any circuit with polynomially many two-qubit Matchgates.

It is possible to extend the theorem to apply to annihilation and creation operators, yielding the following expressions:

$$U^\dagger a_\mu U = \sum_{\nu=1}^{2n} T_{\mu\nu} c_\nu, \quad U^\dagger a_\mu^\dagger U = \sum_{\nu=1}^{2n} T_{\mu\nu}^* c_\nu, \tag{5.6}$$

where $T_{\mu\nu} = \frac{1}{2}(R^T_{2\mu-1,\nu} + iR^T_{2\mu,\nu})$ where $.^T$ is the matrix transpose.

We can use theorem 5.1.1 to simulate (as per our definition at the start of this chapter) a matchgate circuit using the following equation:

$$\langle \mathbf{0}|U^\dagger c_{i_1} c_{i_2} ... c_{i_d} U|\mathbf{0}\rangle = \sum_{\nu_1 \neq \nu_2 \neq ... \nu_d = 1}^{2n} R_{i_1,\nu_1} R_{i_2,\nu_2} ... R_{i_d,\nu_d} \langle \mathbf{0}|c_{\nu_1} c_{\nu_2} ... c_{\nu_d}|\mathbf{0}\rangle. \tag{5.7}$$

This equality is obtained by inserting $UU^\dagger = I$ between each Majorana operator, and applying theorem 5.1.1 to each $U^\dagger c_\mu U$ that appears. We refer to the right hand side of this expression as the "Heisenberg sum", and with the Heisenberg technique we compute the sum explicitly to obtain the solution.

As mentioned in Section 5.1.1, the use of this technique is restricted to observables that are decomposable into a polynomial sum of products of Majorana or fermionic operators (or equivalently a polynomial sum of Pauli strings). If the observable is expressible as a product of rank d in the Majorana operators, the above sum will be $\mathcal{O}(n^d)$ sized and hence computable in polynomial time if d does not increase with n. The overall run-time of the Heisenberg technique for a Matchgate circuit on n qubits and an observable of rank D is $\mathcal{O}(n^D \text{poly}(n))$ [14].

We demonstrate the method in a short example below. The following steps will show how the expectation value of the Pauli Z operator acting on the second and third qubit can be calculated. First of all, using the Jordan-Wigner representation of the Majorana operators, we have the operator $Z_2 \otimes Z_3$ can be expressed as $-c_3c_4c_5c_6$. From there we can use theorem 5.1.1 to derive

$$\begin{aligned}\langle Z_2Z_3\rangle &= \langle \mathbf{0}|U^\dagger Z_2Z_3U|\mathbf{0}\rangle \\ &= \langle \mathbf{0}|U^\dagger(-c_3c_4c_5c_6)U|\mathbf{0}\rangle \\ &= \langle \mathbf{0}| - (U^\dagger c_3U)(U^\dagger c_4U)(U^\dagger c_5U)(U^\dagger c_6U)|\mathbf{0}\rangle \\ &= \sum_{\nu_1\neq\nu_2\neq\nu_3\neq\nu_4=1}^{2n} R_{3,\nu_1}R_{4,\nu_2}R_{5,\nu_3}R_{6,\nu_4}\langle \mathbf{0}| - c_{\nu_1}c_{\nu_2}c_{\nu_3}c_{\nu_4}|\mathbf{0}\rangle,\end{aligned} \tag{5.8}$$

where in the last step we have applied Theorem 5.1.1 with $R = \exp(4h)$ where h is the matrix of coefficients associated to the Hamiltonian of U. The sum runs over all sets of indices where no index is repeated. $\langle \mathbf{0}| - c_{\nu_1}c_{\nu_2}c_{\nu_3}c_{\nu_4}|\mathbf{0}\rangle$ can be computed explicitly. This concludes the example.

So far we have only required our simulations to work for the all-zero $|\mathbf{0}\rangle$ fiducial state. It is possible to extend the Heisenberg simulation method to allow as fiducial state any product state, or even a restricted class of tensor product states. This is proven in the following Proposition.

Proposition 5.1.1. *The Heisenberg technique remains efficiently simulable for any fiducial state which is a tensor product input state $|\Psi_{in}\rangle = |\phi_1\rangle \otimes ... \otimes |\phi_k\rangle$ as long as each $|\phi_i\rangle$ involves up to $\mathcal{O}(log(n))$ qubits.*

Proof. We will first prove the above proposition for computational basis states and arbitrary product states, as these are special cases of the tensor product states considered in the proposition. For a computational basis state input the addition of a product of Majorana operators $C_{\text{input}} = c_{2j_1-1}...c_{2j_w-1}$ acting on the all zero state $|\mathbf{0}\rangle$ is able to generate any computational basis state with Hamming weight w at the cost of n^{2w} additional sums in the Heisenberg sum. Using the Jordan-Wigner form of the Majorana operators (see Eq. (5.5)) we have

$$\langle \mathbf{0}|C_{\text{input}}^\dagger C_{\text{circuit}}C_{\text{input}}|\mathbf{0}\rangle = \prod_i^n \langle 0|P_i^{\text{input}\dagger}P_i^{\text{circuit}}P_i^{\text{input}}|0\rangle, \tag{5.9}$$

where C_{circuit} is a product of Majorana operators arising from the circuit itself.

Next we consider the case of arbitrary product state inputs $|\Psi_{in}\rangle = |\phi_1\rangle|\phi_2\rangle...|\phi_n\rangle$ where we omit the tensor product sign for brevity. To obtain an arbitrary product input state we introduce an ancilla qubit in the $|+\rangle$ state and apply the procedure of [49]:

1. Use the G(H,H) gate and single-qubit Z rotations (These are allowed matchgates) on the $|+\rangle$ ancilla and the neighbouring qubit line to prepare the first qubit in the $|\phi_1\rangle$ state;

2. Use the fermionic swap gate G(Z,X) to swap $|\phi_1\rangle$ all the way out to the final qubit line;

3. Repeat steps 1 and 2 sequentially to prepare $|\phi_i\rangle$ and move it out to qubit line i until all qubits are in the desired state.

After following this procedure the ancilla can be ignored and the original matchgate circuit can be executed on the product state. To prove this procedure can be simulated efficiently, note $|+\rangle|0^{\otimes n}\rangle = (1 + a_0^\dagger)|0^{\otimes n+1}\rangle$ (again omitting the normalization constant). One can simulate each term on the right hand side of this expression individually:

$$\langle 0^{\otimes n+1}|\alpha_j^\dagger C_{\text{circuit}}\alpha_j|0^{\otimes n+1}\rangle = (\langle 0|\alpha_j^\dagger \alpha_j|0\rangle)\prod_i^n \langle 0^{\otimes n}|P_i^{\text{circuit}}|0^{\otimes n}\rangle, \tag{5.10}$$

where $\alpha_j \in \{1, a_0^\dagger\}$[1]. This increases the overall run-time of the algorithm by a factor of 2. The gates used in step 1 & 2 can be absorbed into the matchgate circuit.

It remains to prove arbitrary tensor product states $|\Psi_{\text{in}}\rangle = |\phi_1\rangle \otimes ... \otimes |\phi_k\rangle$ where each $|\phi_i\rangle$ involves at most $\mathcal{O}(\log(n))$ qubits are allowable fiducial states in classically efficient matchgate simulations. To simulate some tensor product input states $|\Psi\rangle = |\phi_1\rangle \otimes ... \otimes |\phi_k\rangle$ we generalize Eq. (5.9):

$$\langle \Psi|C_{\text{circuit}}|\Psi\rangle = \prod_i^k \langle \phi_i|P_i^{\text{circuit}}|\phi_i\rangle. \tag{5.11}$$

P_i^{circuit} is now itself a tensor product of Paulis acting on the qubit lines part of ϕ_i. By restricting the size of $|\phi\rangle$ to at most $\mathcal{O}(\log(n))$ qubits the matrices involved in Eq. 5.11 are poly(n) sized and hence the overall expression is computable in polynomial time. □

5.1.3 Wick's simulation

With the Heisenberg technique we end up explicitly computing the Heisenberg sum. The size of the sum grows exponentially if the rank of the observable scales as $\mathcal{O}(n)$, which is renders the method classically inefficient. Terhal & DiVincenzo demonstrated Wick's Theorem for ordinary operator products can be used to compute a Heisenberg sum more efficiently [12]. Wick's theorem allows us to write a Heisenberg sum as the Pfaffian of an antisymmetric square matrix O[2], which we can construct using a lookup table. The Pfaffian of an antisymmetric matrix has the property that $\text{Pf}(A)^2 = \det(A)$, and since the determinant of O can be computed efficiently, the Pfaffian can too. As we will see, O can be constructed efficiently too. For brevity we refer the reader to [12, 30] for a full discussion of Wick's Theorem. Here we describe the process of constructing O, starting from a Heisenberg sum. First, we define the block diagonal matrix

$$H = \bigoplus_{l=1}^n \begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix}. \tag{5.12}$$

[1]This equation already takes into account that the cross terms arising from $\langle 0^{\otimes n+1}|(1+a_0^\dagger)^\dagger ... (1+a_0^\dagger)|0^{\otimes n+1}\rangle$ evaluate to zero.

[2]The Pfaffian and O arise from the theory of perfect matchings in graphs. For a discussion see Chapter 3 of [14]

To find the entry $O_{i,j}$, look at the subscript of the ith and jth Majorana operator in the Heisenberg sum. Find the matrices R which have this subscript as their column index. We shall refer to these matrices as $R(i)$ and $R(j)$. Compute the matrix product $R(i)HR^T(j)$. Note down the row index of $R(i)$ and $R(j)$, say α and β: $O_{i,j}$ is equal to $(R(i)HR^T(j))_{\alpha,\beta}$. The entries of O can be populated by extracting the relevant matrix elements of the stored $R(i)HR^T(j)$. Wick's theorem tells us to repeat this for each pair of Majorana operators in the Heisenberg sum. Since O is anti-symmetric only the pairs $i < j$ have to be found explicitly. The dimension of O for a purely Matchgate circuit and observable of rank D is D. As the dimension of R (and H) is $2n$ one can compute the matrix product $R(i)HR^T(j)$ in poly(n) time. The cost of computing the determinant of O is $\mathcal{O}(D^3)$ meaning the time complexity of the Wick's simulation method is $\mathcal{O}(D^3\text{poly}(n))$.

Understanding the above method might be aided by the following example based on the Heisenberg sum from Eq. (5.8). Consider the element $O_{1,3}$, this element is related to the first and third Majorana operators: c_{ν_1} and c_{ν_3}. Recall or compute (RHR^T). The l_1 (l_3) index associated to ν_1 (ν_3) is 3 (5), giving $O_{1,3} = (RHR^T)_{3,5}$. Using the anti-symmetric property of O we can set $O_{3,1}$ equal to $-O_{1,3}$. We can repeat this process to find every element of O. Finally we have $\langle Z_2 Z_3\rangle = \sqrt{\det(O)}$. This concludes the example.

For a pure Matchgate circuit each R is the same, so RHR^T only has to be computed once and stored in memory. This does not hold for circuits which have additional features, including but not limited to computational basis states as the fiducial state, or mid-circuit measurements. For these circuits it may be useful to construct a lookup table to keep track of the different matrix products to be computed, and avoid repeatedly computing the same matrix products. The gain in efficiency is hidden in the poly(n) contribution to the time complexity.

An important caveat to Wick's simulation is that the Majorana operators in the final Heisenberg sum may only act on the all-zero fiducial state $|\mathbf{0}\rangle$ [12]. Consequently we have the following lemma:

Lemma 5.1.1. *Matchgate circuits with computational basis or product input states are efficiently classically simulable using Wick's simulation. Arbitrary tensor product input states $|\Psi_{in}\rangle = |\phi_1\rangle \otimes ... \otimes |\phi_k\rangle$ where each $|\phi_i\rangle$ involves at most $\mathcal{O}(log(n))$ qubits are not.*

Proof. Both Eq. (5.9) and Eq. (5.10) consider expectation values over the vacuum state $|0^{\otimes n}\rangle$, whilst in Eq. (5.11) arbitrary multi-qubit states are considered. $\square$

Remark. *The scaling for computational basis states increases to $\mathcal{O}((D+2w)^3 poly(n))$ where w is the Hamming weight of the computational basis state. We will explain the method below.*

Remark. *Using product state inputs increases the size of matrix O by at most 2, when simulating the $a_0^\dagger|0^{n+1}\rangle$ state. Otherwise the additional computational cost is captured by the increased circuit depth used in the procedure of [49].*

5.1.4 The SWAP gadget - proof of theorem 5.0.1

The SWAP gadget [48] is a tool inspired by the concept of magic states introduced in [50] which allows the realization of a SWAP gate using alternative resources.

The gadget achieves this using two key features: *magic states* and *adaptive measurements.* These two features by themselves are allowable additions to classically simulable matchgate circuits, however together yield the much more powerful SWAP gadget [30], shown in Fig. 5.1. In this section we will first present the methods by which the features individually are efficiently simulable, and followed by the SWAP gadget method. We conclude with the proof of theorem 5.0.1.

Let us consider how to simulate Matchgate circuits supplemented with magic states. Magic states for Matchgate computation are states that A) cannot be generated from a particular restricted gate set, and B) can be moved across qubit lines "freely" by gates from the same restricted gate set. Consequently, we have the following theorem:

Theorem 5.1.2 ([48])**.** *Any pure fermionic state which is non-Gaussian is a magic state for matchgate computations*

An example of a magic state is the following: $\frac{1}{2}(|0000\rangle+|0011\rangle+|1100\rangle+|1111\rangle)$, which is the product of two $|\Phi^+\rangle$ Bell states. Magic states are a special type of tensor product input state, meaning Matchgate circuits supplemented with magic states are classically simulable through Lemma 5.1.1, but not using Wick's simulation due to Lemma 5.1.1.

Next, we consider adaptive measurements. Adaptive measurements are measurements in the computational basis that do not necessarily take place at the end of a circuit (we refer to these as "intermediate measurements"), and the outcome of which conditions a gate on a different qubit. Such intermediate measurements can be included in Matchgate simulations by calculating the conditional probability of obtaining a particular final measurement given the outcome of the intermediate measurements.

To demonstrate the procedure by which one can simulate matchgate circuits containing adaptive measurements, let us follow the notation of [30]. The probability of obtaining a bitstring x after measuring some matchgate circuit is $p(x)$. For a circuit containing a single intermediate measurement we have the joint probability $p(x, y_1)$ of obtaining the intermediate measurement outcome y_1 and final outcome x, or in the case of J intermediate measurements we have $p(x, y_J, ..., y_1)$. We can calculate these marginal and joint probabilities as follows. Let U_1 be the subset of gates that affect the first intermediate measurement and $\Pi(y_1)$ the projector of y_1, then

$$p(y_1) = \langle\psi|U_1^\dagger \Pi(y_1) U_1|\psi\rangle, \tag{5.13}$$

which can be computed efficiently using for example Wick's simulation. The general form of the joint probabilities is

$$p(y_{i+1}, y_i, ..., y_1) = \langle\psi|(\prod_{k=1}^{i} U_k^\dagger \Pi(y_k))U_{i+1}^\dagger \Pi(y_{i+1})U_{i+1}(\prod_{k=1}^{i} \Pi(y_k)U_k)|\psi\rangle, \tag{5.14}$$

where i runs from 1 to J. When using Wick's simulation, the dimension of the matrix O is $D+4J$ where D is twice the length of the bitstring x and J the number of intermediate measurements. The cost of computing the determinant is hence $(D+4J)^3$, i.e. polynomial in the number of intermediate measurements. Consequently the method remains classically simulable for poly(n) intermediate measurements. To enable gates conditioned on the intermediate measurement note that each section of the matchgate circuit now depends on the outcome of the measurements in the previous sections; $U_k \to U_k(y_{k-1}, ..., y_1)$ cf Eq. (5.14).

Critically, we must recognize that in order to obtain $p(x)$ we must sum over all possible intermediate measurement outcomes: $p(x) = \sum_{y_J,...,y_1} p(x, y_J, ..., y_1)$. Generally, there are exponentially many combinations of such outcomes possible and therefore it is not feasible to simulate circuits containing adaptive measurements in the strong sense (as per Definition 2.0.1). Instead, we introduce the procedure of [12] to sample from p(x) instead (which corresponds to weak simulation as per Definition 2.0.3):

1. Simulate the circuit up to the first measurement to obtain y_1, p(y_1) using Eq. (5.13);
2. classically sample one outcome y_1 from p(y_1) and fix this for the remainder of the procedure;
3. compute p(y_1, y_2) using Eq. (5.14), from which p($y_2|y_1$) = p(y_2, y_1)/p(y_1) can be calculated;
4. classically sample one outcome y_2 from p($y_2|y_1$);
5. repeat the previous steps for each adaptive measurement up until the final measurement outcome x is sampled.

We now have all the ingredients for the SWAP gadget, shown in Fig. 5.1. The procedure by [48] is as follows. First, one copy of a magic state is moved to in-between the two qubit lines to be swapped. Through the Bell measurements the target states are available on qubit lines 2 and 3, which are corrected up to local Pauli equivalence immediately after. The Z corrections are done using the G(Z,Z) = Z⊗I matchgate. For the X corrections first a $|0\rangle$ ancilla is moved in using fSWAPs (G(Z,X)), then a G(X,X) gate acts on it and the target line, after which G(-Z,X) gates are used to move the ancilla out again. The $|\psi\rangle$ and $|\phi\rangle$ states have now been swapped.

As evident from Fig. 5.1, to deterministically implement a SWAP gate using the SWAP gadget, one magic state is consumed and four adaptive measurements are made. In order to classically simulate matchgate circuits containing SWAP gadgets we turn to Wick's simulation. As the magic states cannot be simulated within the framework of Wick's simulation, we instead consider them as a superposition of computational basis states so that we can apply Eq. (5.9). Each magic state is made up of at best 4 computational basis states, so for a circuit containing N SWAP gadgets $4^N \times 4^N$ cross terms are to be computed. Through symmetry we can reduce this to $\frac{1}{2}4^N(4^N - 1)$. Each cross term can be expressed as a Heisenberg sum and consequently compute using Wick's simulation (the lookup table for these computations is printed in [30]).

The dimension of each O is at most $(D + 16N + 8N)$, where the $16N$ comes from the adaptive measurements and $8N$ from the maximum combined Hamming weights of the computational basis states representing the Magic states. As a result the overall scaling of the algorithm is $O(4^{2N}(D + 24N)^3\text{poly}(n))$ or equivalently $\mathcal{O}(4^{2N}\text{poly}(n, D, N))$. Due to the use of adaptive measurements the SWAP gadget corresponds to the weak level of simulation. This proves theorem 5.0.1.

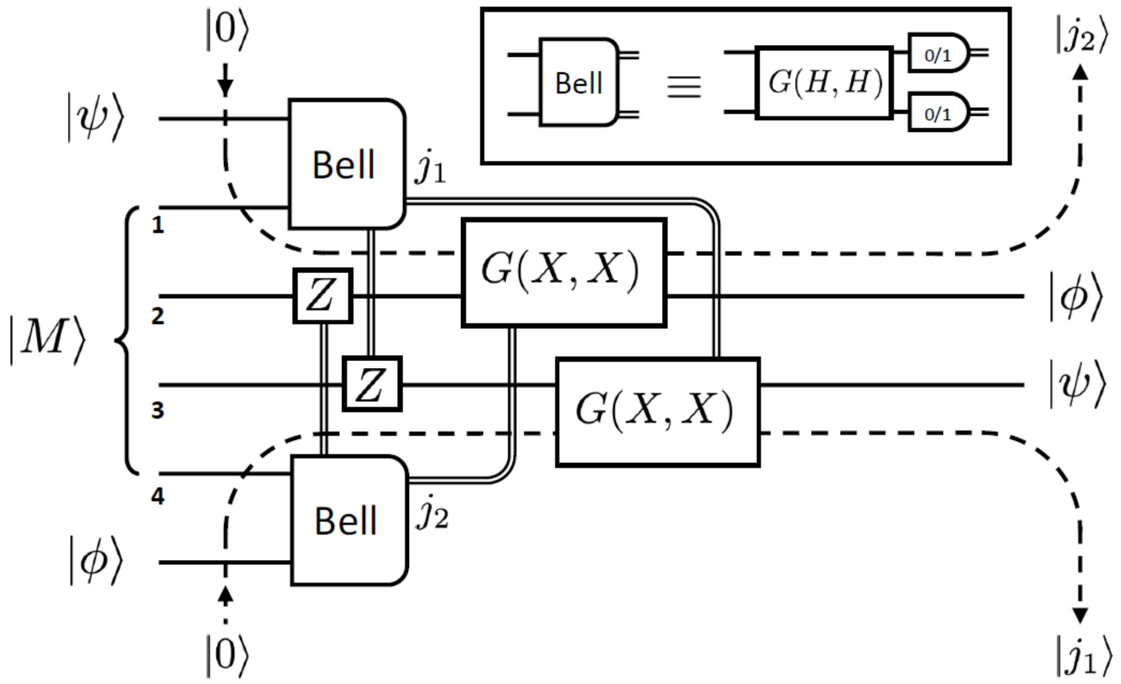


Figure 5.1: A circuit diagram of the SWAP gadget. $|M\rangle$ is the magic state $|\Phi^+_{1,3}\rangle|\Phi^+_{2,4}\rangle$ which is equivalent to the magic state in the text up to a fermionic swap gate. The Bell measurements can be implemented by applying the G(H,H) matchgate, followed by two local computational basis measurements. The outcomes of the Bell measurements are used to apply the local Pauli corrections, connected via solid double lines. The dashed lines represent ancillas used in the Pauli X corrections. Reproduced from [48].

5.1.5 SWAP gate decomposition for Matchgate circuits

The SWAP gadget is not a suitable simulation method for us to address research question 3, as it belongs to the weak level of simulation due to the use of adaptive measurements. The core idea underpinning our simulation algorithm is to express the SWAP gate as a sum of matchgates only. This circumvents the use of adaptive measurements. Since matchgate only circuits are classically simulable, one can calculate the expectation value of an observable for a circuit containing SWAP gates by considering the various matchgate only circuits arising from the matchgate decomposition of the SWAP gate. In this section we explain the basic workings of our method.

To decompose the SWAP gate into matchgates we use the Schmidt decomposition, which in the Pauli basis is as follows:

$$SWAP = \frac{1}{2}(I \otimes I + X \otimes X + Y \otimes Y + Z \otimes Z). \tag{5.15}$$

It has a Schmidt number of 4. These terms correspond to the G(I,I), G(X,X), G(-X,X) and G(I,-I) matchgates respectively, as required. It will be convenient to decompose these matchgates into the Majorana operators. For further convenience, let us define the "Pauli products" $P_A := A \otimes A$, where $A \in \{I, X, Y, Z\}$ are the Pauli matrices. In terms of Majorana operators we have

$$\begin{aligned} P_I^j =& I \\ P_X^j =& -ic_{2j}c_{2j+1} \\ P_Y^j =& ic_{2j-1}c_{2j+2} \\ P_Z^j =& -c_{2j-1}c_{2j}c_{2j+1}c_{2j+2}, \end{aligned} \tag{5.16}$$

where P^j indicates the operator is acting on qubit j and $j+1$. Trivially the index j is irrelevant for P_I. Note we can trivially express the SWAP gate in terms of the Pauli products;

$$SWAP = \frac{1}{2}(P_I + P_X + P_Y + P_Z). \tag{5.17}$$

Before we move on let us first state some product and commutation relations of the Pauli products that may be readily derived from the Pauli matrices:

$$\begin{aligned} P_a = P_a^\dagger &\quad \forall a \in \{I, X, Y, Z\}, \\ P_aP_I = P_IP_a = P_a &\quad \forall a \in \{I, X, Y, Z\}, \\ P_aP_b = -P_c &\quad \forall a \neq b \neq c \quad (\text{+permutations}) \in \{X, Y, Z\}, \\ P_aP_a = P_I &\quad \forall a \in \{I, X, Y, Z\}, \\ [P_a, P_b] = 0 &\quad \forall a, b \in \{I, X, Y, Z\}. \end{aligned} \tag{5.18}$$

The goal is to develop a simulation method which takes as input a circuit containing both Matchgates and a few SWAP gates, a set of parameters $\boldsymbol{\theta}$ and an observable f. Consider the following circuit: a block of matchgates U, a SWAP gate, followed by another block of matchgates V (see Fig. 5.2). We are interested in calculating the expectation value of some observable f:

$$\langle \mathbf{0}|(V(SWAP)U)^\dagger f(V(SWAP)U)|\mathbf{0}\rangle,$$

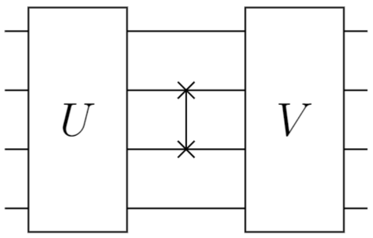


Figure 5.2: Diagram of a matchgate circuit containing a SWAP gate. U and V are Gaussian operators, which are separated by a SWAP gate.

which, up to a normalization constant, can be expressed as

$$\begin{aligned}\langle\mathbf{0}|(V(P_I+P_X+P_Y+P_Z))U)^\dagger f(V(P_I+P_X+P_Y+P_Z)U)|\mathbf{0}\rangle = \\ \langle\mathbf{0}|(VP_IU)^\dagger f(VP_IU)|\mathbf{0}\rangle + \langle\mathbf{0}|(VP_IU)^\dagger f(VP_XU)|\mathbf{0}\rangle + \\ \langle\mathbf{0}|(VP_IU)^\dagger f(VP_YU)|\mathbf{0}\rangle + \langle\mathbf{0}|(VP_IU)^\dagger f(VP_ZU)|\mathbf{0}\rangle + \\ \langle\mathbf{0}|(VP_XU)^\dagger f(VP_IU)|\mathbf{0}\rangle + \langle\mathbf{0}|(VP_XU)^\dagger f(VP_XU)|\mathbf{0}\rangle + \\ \vdots \\ \langle\mathbf{0}|(VP_ZU)^\dagger f(VP_YU)|\mathbf{0}\rangle + \langle\mathbf{0}|(VP_ZU)^\dagger f(VP_ZU)|\mathbf{0}\rangle = \\ \sum_{a,b}^{\{I,X,Y,Z\}} \langle\mathbf{0}|(VP_aU)^\dagger f(VP_bU)|\mathbf{0}\rangle,\end{aligned} \tag{5.19}$$

using the linearity of the inner product (the normalization constant is $\frac{1}{4}$). This yields us 16 terms where the operators either side of the observable are not necessarily each others conjugate, but are purely matchgate circuits. Four of these terms are expectation values, and the remaining twelve are inner products. Provided the observable f is Hermitian it is clear the inner product terms come in pairs that are the complex conjugates of each other, meaning $S_4 = 4(4+1)/2 = 10$ terms are to be computed overall. For a circuit containing N SWAP gates, $4^N(4^N+1)/2$ terms have to be computed. The challenge is to find a simulation algorithm that can compute both expectation values as well as inner products. We prove this is possible with both the Heisenberg technique, and Wick's simulation, yielding the Divide & Heisenberg and Divide & Wick methods.

5.1.6 Divide & Heisenberg - proof of theorem 5.0.2

In order to prove theorem 5.0.2 we must show we can express the various terms appearing Eq. (5.19) as Heisenberg sums. We will demonstrate it is possible for the generalized case of a Matchgate circuit containing N SWAP gates. The inner product to be computed in the simulation of such circuits is $\langle\phi|U_a^\dagger f U_b|\phi\rangle$, where $U_a = \left(\prod_{i=1}^N U_i P_{a_i}\right)U_0 = U_N P_{a_N} U_{N-1} ... P_{a_1} U_0$ and similarly $U_b = \left(\prod_{j=1}^N U_j P_{b_j}\right)U_0$. The P operators are Pauli products which arise due to the decomposition of the SWAP gates into these Pauli products. The only difference between U_a and U_b are the Pauli products P_a and P_b. Both U_a and U_b are matchgate circuits. Note the index of the qubits where the ith SWAP gate acted before being decomposed is implicitly included in the label a_i and b_i. For convenience let us introduce the following notation: $U_{b:l} = \left(\prod_{j=1}^l U_j P_{b_j}\right)U_0$ with $U_{b:0} = U_0$.

Without loss of generality we can take $f = c_1 c_2$, as the below procedure is readily generalized to an observable consisting of arbitrary product of fermionic or Majorana operators. The following procedure demonstrates how to obtain the Heisenberg sum of some inner product $\langle\phi|U_a^\dagger f U_b|\phi\rangle$ where $P_a \neq P_b$:

1. Introduce $U_{b:N}U_{b:N}^\dagger$ before each operator from the observable:
$$\langle\mathbf{0}|U_{a:N}^\dagger c_1 c_2 U_{b:N}|\mathbf{0}\rangle = \langle\mathbf{0}|U_{a:N}^\dagger U_{b:N}U_{b:N}^\dagger c_1 U_{b:N}U_{b:N}^\dagger c_2 U_{b:N}|\mathbf{0}\rangle. \tag{5.20}$$

2. Apply theorem 5.1.1;
$$\langle\mathbf{0}|U_{a:N}^\dagger U_{b:N}U_{b:N}^\dagger c_1 U_{b:N}U_{b:N}^\dagger c_2 U_{b:N}|\mathbf{0}\rangle = \sum_{\nu,\mu}^{2n} R_{1,\nu}^{U_{b:N}} R_{2,\mu}^{U_{b:N}} \langle\mathbf{0}|U_{a:N}^\dagger U_{b:N} c_\nu c_\mu|\mathbf{0}\rangle, \tag{5.21}$$
where $R^{U_{b:N}} = e^{4h_{U_{b:N}}}$. From here on all sums are implied to be over $2n$.

3. The product $U_N^\dagger U_N$ in $U_{a:N}^\dagger U_{b:N}$ cancels to give $U_{a:N-1}^\dagger P_{a_N} P_{b_N} U_{b:N-1}$. $P_{a_N} P_{b_N}$ can be simplified by the product identities (Eq. (5.18)), after which we can insert its Majorana operator representation $C_{k_1,\ldots,k_d} = c_{k_1}\ldots c_{k_d}$:
$$\sum_{\nu,\mu} R_{1,\nu}^{U_{b:N}} R_{2,\mu}^{U_{b:N}} \langle\mathbf{0}|U_{a:N}^\dagger U_{b:N} c_\nu c_\mu|\mathbf{0}\rangle = \sum_{\nu,\mu} R_{1,\nu}^{U_{b:N}} R_{2,\mu}^{U_{b:N}} \langle\mathbf{0}|U_{a:N-1}^\dagger C_{k_1,\ldots,k_d} U_{b:N-1} c_\nu c_\mu|\mathbf{0}\rangle. \tag{5.22}$$

4. Introduce $U_{b:N-1}U_{b:N-1}^\dagger$ before each operator in $C_{k_1,\ldots,k_d}$ and again apply theorem 5.1.1, giving
$$\sum_{l_1,\ldots,l_d,\nu,\mu} R_{k_1,l_1}^{U_{b:N-1}} \ldots R_{k_d,l_d}^{U_{b:N-1}} R_{1,\nu}^{U_{b:N}} R_{2,\mu}^{U_{b:N}} \langle\mathbf{0}|U_{a:N-1}^\dagger U_{b:N-1} C_{l_1,\ldots,l_d} c_\nu c_\mu|\mathbf{0}\rangle. \tag{5.23}$$

5. Repeat step 3 and 4 until all the Pauli products have been resolved no unitaries remain in the inner product:
$$\sum_{\substack{q_1,\ldots,q_d,\\ \ldots,\\ l_1,\ldots,l_d,\\ \nu,\mu}} R_{p_1,q_1}^{U_{b:0}} \ldots R_{p_d,q_d}^{U_{b:0}} \ldots R_{k_1,l_1}^{U_{b:N-1}} \ldots R_{k_d,l_d}^{U_{b:N-1}} R_{1,\nu}^{U_{b:N}} R_{2,\mu}^{U_{b:N}} \langle\mathbf{0}|C_{q_1,\ldots,q_d} \ldots C_{l_1,\ldots,l_d} c_\nu c_\mu|\mathbf{0}\rangle. \tag{5.24}$$

The final expression is the Heisenberg sum of an inner product of the form $\langle\phi|U_a^\dagger f U_b|\phi\rangle$ where $P_a \neq P_b$. To prove theorem 5.0.2 it remains to calculate the time complexity.

With the Heisenberg technique the sum is computed explicitly. For an observable of rank D steps 1 and 2 generate D sums. In step 3 we introduce the general notation for a product of Majorana operators $C_{p_1,\ldots,p_d}$, where the sub-index d refers to the degree of the Pauli product $P_{a_i}P_{b_i}$. Based Eq. (5.16) when $P_{a_i} = P_{b_i}$ we have d=0, for $P_{a_i}P_{b_i} = P_X$ or P_Y d=2 and finally if $P_{a_i}P_{b_i} = P_Z$ we have $d = 4$. The overall size of the sum is $n^{D+\sum_j^N d_j}$ for a circuit containing N SWAP gates, which is maximal when all d_j's evaluate to 4. In this case the size of the sum is n^{D+4N}. Considering the number of inner products that have to be computed is $\frac{1}{2}4^N(4^N+1)$ for a matchgate circuit containing N SWAP gates, and the number of sums for each computation is at most n^{D+4N}, the cost of the algorithm scales as $\mathcal{O}(4^{2N}n^{D+4N})$. This proves theorem 5.0.2. Clearly the complexity is polynomial in the number of qubits n as long as the number of SWAP gates N and the rank of the observable D are independent of n. The algorithm may find use cases when the number of SWAP gates is $O(1)$ and the observable considers few qubits.

5.1.7 Divide & Wick - proof of theorem 5.0.3

We have demonstrated the inner products arising from decomposing SWAP gates into a sum of Matchgates can be expressed as a Heisenberg sum. We have seen Wick's simulation is a method whereby one avoids explicitly computing the Heisenberg sum. Consequently, we can readily improve the efficiency of theorem 5.0.2 by using Wick's simulation. The overall recipe for computing O remains the same, but there are two key differences. Firstly, the dimension of O increases, specifically with the number of SWAP gates. Secondly, the lookup table is bigger to accommodate the various matrix products arising from the R matrices in the Heisenberg sum. A lookup table for a generic Heisenberg sum such as Eq. (5.24) is given in Table 5.2

	c_{q_β}	$\cdots$	c_{k_β}	c_ν	c_μ
c_{q_α}	$(R^{U_{b:0}}HR^{U_{b:0}T})_{p_\alpha,p_\beta}$	$\cdots$	$(R^{U_{b:0}}HR^{U_{b:N-1}T})_{p_\alpha,l_\beta}$	$(R^{U_{b:0}}HR^{U_{b:N}T})_{p_\alpha,1}$	$(R^{U_{b:0}}HR^{U_{b:N}T})_{p_\alpha,2}$
$\vdots$	X	$\ddots$			
c_{k_α}	X	X	$(R^{U_{b:N-1}}HR^{U_{b:N-1}T})_{l_\alpha,l_\beta}$	$(R^{U_{b:N-1}}HR^{U_{b:N}T})_{l_\alpha,1}$	$(R^{U_{b:N-1}}HR^{U_{b:N}T})_{l_\alpha,2}$
c_ν	X	X	X	X	$(R^{U_{b:N}}HR^{U_{b:N}T})_{1,2}$
c_μ	X	X	X	X	X

Table 5.2: Lookup table used to construct the matrix O for a generic Heisenberg sum given by Eq. (5.24). X indicates such combinations of Majorana operators do not appear. $.^T$ denotes the transpose operation w.r.t. the computational basis. Additional rows and columns can be added in the row and column denoted by three dots to accommodate further subscripts introduced by additional SWAP gates.

Based on the number of sums in a generic Heisenberg sum (Eq. (5.24)) it can be inferred that the dimension of O is $D+\sum_j^N d_j$, up to $D+4N$ where D is the rank of the observable. For each SWAP gate introduced the dimension of O increases by up to four, but the size of the lookup table only grows if a partition is introduced in a block of unitaries by the SWAP gate. Wick's simulation can also be applied to the different types of input state discussed in Lemma (5.1.1). For computational basis state inputs, the dimension of O increases by w where w is the Hamming weight of the basis state. For a product or tensor product input state the dimension of O increases by at most one for the $a_0^\dagger$ coefficient, however the overall cost of the simulation increases by a factor of two as two circuits have to be evaluated, one with each superposition component of the $|+\rangle$ ancilla.

The cost of computing the determinant of a matrix of dimension M is M^3. The dimension of O is at worst $D + 4N$, meaning the complexity of computing each inner product is $(D+4N)^3$ which is polynomial in D and N. With the D&C scheme $\frac{1}{2}4^N(4^N+1)$ of such inner products have to be computed, giving an overall scaling of $O(4^{2N}(D+4N)^3\text{poly}(n))$ which can be simplified to $\mathcal{O}(4^{2N}\text{poly}(n, D, N))$ or equivalently $\mathcal{O}(4^{2N}\text{poly}(n, N))$ as physically D can be at most $\mathcal{O}(n)$. This proves theorem 5.0.3. Additional costs associated to product state inputs are readily absorbed into the prefactor constant and $\text{poly}(n, D, N)$ term.

5.1.8 Discussion

We have presented two algorithms which provide an alternative method to simulate matchgate circuits containing SWAP gates to the SWAP gadget developed by Hebenstreit et. al [48]. Our algorithms work by decomposing the SWAP gate into a sum of matchgates. As a result each term by itself is a pure matchgate circuit which can be classically simulated in polynomial time. By taking this decomposition a number of cross terms arise which have to be computed: we show that it is possible to compute each cross term in polynomial time too. The number of cross terms however, is exponential in the number of SWAP gates in the circuit. We compare the cost of simulation for our algorithms to that of the SWAP gadget in Table 5.1.

From the table it is evident that Wick's simulation (used in both the SWAP gadget and Divide & Wick) allows for a polynomial algorithm even when the number of qubits measured is $\mathcal{O}(n)$, whilst the Heisenberg technique does not. Furthermore, the D&C methods are of the semi-strong level of simulation for observables which can be expressed as a product of fermionic and/or Majorana operators acting on n_q qubits. This includes all projectors as

$$|0\rangle\langle 0| = aa^\dagger \qquad\qquad |1\rangle\langle 1| = a^\dagger a, \tag{5.25}$$

where the qubit index is omitted. Pauli string observables also meet this criteria, as the equivalent product of Majorana operators can be obtained through its Jordan-Wigner transformation (see Eq. (5.5)). One can also plug in the above observables directly into the SWAP gadget. This does not promote it to the semi-strong level of simulation, as one still has to sample over all the possible adaptive measurement outcomes.

One practical difference between the Divide & Wick and the SWAP gadget methods is the cost of increasing the distance between the two qubit lines on which the SWAP gate acts. As is evident from Fig. 5.1, qubit lines 2 and 3 of the magic state $|M\rangle$ can be separated by any number of qubits as there are no matchgates acting across them (recall matchgates can act on nearest neighbour qubits only). We only require that the outcome of any intermediate measurement can condition a matchgate on any pair of qubits. For the Divide & Wick method however, we assumed the SWAP gate was acting on nearest-neighbour qubits when decomposing the tensor product of Paulis into a product of Majorana operators. For a SWAP gate acting on two qubits separated by d qubit lines, this decomposition becomes a product of $2(d + 1)$ operators for P_X and P_Y (cf Eq. (5.16)) whilst P_Z remains a product of just four operators. This change only affects the size of the matrix O as follows: for a set of N SWAP gates $G \in \{G_i\}^N$, each acting on qubits which are d_i qubit lines apart, the dimension of O is at worst $(D+\max(2\sum_i^N(d_i+1), 4N))$. Assuming $\max(2\sum_i^N(d_i + 1), 4N) = 2\sum_i^N(d_i + 1)$, the scaling in the Divide & Wick theorem

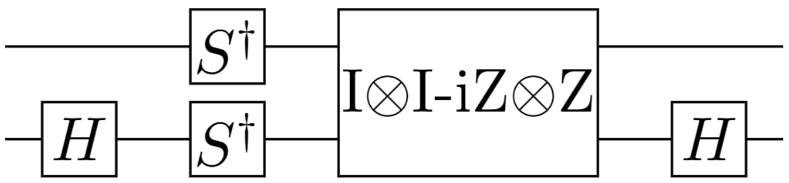


Figure 5.3: A decomposition of a CNOT gate into a tensor product two qubit gate, supplemented by single qubit gates. The $S^\dagger$ gate is the phase gate, given by diag(1, $-i$). Adopting the form of Eq. (5.26) we have $\alpha_0 = \frac{1}{\sqrt{2i}}$ and $\alpha_1 = \frac{-i}{\sqrt{2i}}$.

becomes $\mathcal{O}(4^{2N}(D + 2\sum_i^N (d_i + 1))^3 \text{poly}(n))$. This teaches us that qubit swapping across a distance is a fundamentally quantum feature. Crucially, the exponential scaling in the number of SWAP gates remains unaffected.

Whilst the SWAP gadget is more suited to circuits that contain SWAP gates acting on distant qubits, the Divide & Wick method is still capable of a stronger level of simulation. We associate this to the lack of adaptive measurements in our alternative implementation of the SWAP gate. Consequently, it has more practical use-cases, namely those where the expectation value of some observable is considered as the output of some model. This idea is the foundation of many near term algorithms, such as Variational Quantum Algorithms [51, 5], and as a result our new algorithm is more applicable than the SWAP gadget.

5.2 Circuit Partitioning

The run-time scaling of our Matchgate algorithms are likely too expensive for us to properly probe the relation between the learning performance and computational hardness of a Matchgate model. For this reason we consider here an entirely different simulation framework as an alternative candidate. We call the approach *Circuit Partitioning*, and will use it to prove theorem 5.0.4.

The Circuit Partitioning formalism uses the *gate decomposition* technique to decompose (or "cut") two- (or multi-) qubit gates into a sum of tensor products of single qubit unitaries. This can be done using the operator-Schmidt decomposition:

$$U^{ij} = \sum_a^{n_s} \alpha_a U_a^i \otimes U_a^j, \tag{5.26}$$

where U^{ij} is a unitary acting on qubits i and j, α_a a constant and U^i (U^j) a single qubit unitary acting on qubit i (j). The number of terms in the sum n_s is also known as the Schmidt number. All two-qubit gates have a Schmidt number of four or smaller: the CNOT gate is an entangling gate with a Schmidt number of 2 [52].

The original CNOT operator-Schmidt decomposition by Nielsen is as follows:

$$\text{CNOT} = (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)/\sqrt{2}, \tag{5.27}$$

which involves projective measurements. The decomposition in the Pauli basis uses Hadamard and Phase gates along with a two-qubit Pauli tensor product gate, and is shown in Fig. 5.3.

Consider a quantum device where the set of qubits is partitioned into blocks A and B, with each qubit in one block only. Let U be a quantum circuit that can be decomposed as blocks of unitaries U^A, U^B and U^{AB} acting on A, B and AB respectively;

$$U = (U_1^A \otimes U_1^B) U^{AB} (U_2^A \otimes U_2^B). \tag{5.28}$$

By inserting the operator-Schmidt decomposition of U^{AB} we get a sum of circuits that act on A and B independently;

$$\sum_a^{n_s} \alpha_a (U_1^A \otimes U_1^B) U_a^A \otimes U_a^B (U_2^A \otimes U_2^B) = \sum_a^{n_s} \alpha_a (U_1^A U_a^A U_2^A) \otimes (U_1^B U_a^B U_2^B). \tag{5.29}$$

This expression can readily be generalized for a unitary U with L repeating layers of $U^A \otimes U^B$ and U^{AB}:

$$U = \prod_i^L (U_i^A \otimes U_i^B) U_i^{AB} = \sum_{a_1, a_2, \ldots, a_L}^{n_s} \alpha_{a_1} \alpha_{a_2} \ldots \alpha_{a_L} (\prod_i^L U_i^A U_{a_i}^A \otimes U_i^B U_{a_i}^B). \tag{5.30}$$

Suppose one would like to compute the expectation value of an observable $f = f^A \otimes f^B$ of this circuit U acting on some fiducial state $|\Phi\rangle = |\phi^A\rangle \otimes |\phi^B\rangle$. Using Eq. (5.30) we are able to simplify the computation to a sum of inner products of the partitions independently;

$$\begin{aligned}\langle \Phi | U^\dagger f U | \Phi \rangle = & \sum_{\{a_q\}_{q=1}^L, \{b_q\}_{q=1}^L}^{n_s} \alpha_{a_1}^* \alpha_{a_2}^* \ldots \alpha_{a_L}^* \alpha_{b_1} \alpha_{b_2} \ldots \alpha_{b_L} \\ & \langle \phi^A | (\prod_i^L U_i^A U_{a_i}^A)^\dagger f^A (\prod_j^L U_j^A U_{b_j}^A) | \phi^A \rangle \langle \phi^B | (\prod_p^L U_p^B U_{a_p}^B)^\dagger f^B (\prod_q^L U_q^B U_{b_q}^B) | \phi^B \rangle,\end{aligned} \tag{5.31}$$

where n_s is the Schmidt number of U^{AB}. The advantage of this alternative computation is that the dimensions of the wavefunctions are $2^{|A|}$ and $2^{|B|}$, rather than $2^{|A|+|B|}$, at the expense of having to compute n_s^{2L} inner products. Through the interchange of the a and b subscripts it is evident for each $a_i \neq b_j$ the complex conjugate of that inner product appears in the sum too, meaning we can reduce the number of inner products that are to be explicitly computed to the sum of natural numbers $S_{n_s^L} = n_s^L(n_s^L + 1)/2$.

Generalizing further the above method to the case of multiple partitions yields the following proposition.

Proposition 5.2.1 (Partitioned circuit). *Consider a circuit U on n qubits. Partition the qubits into B sets b_i $\forall i \in [B]$ (where $[B] := 1, 2, ..., B$) such that each qubit appears in one set and one set only. Define an observable that can be decomposed into a tensor product of observables that considers independently each partition $f = \bigotimes_i^B f_i$. We can always express the expectation value of such an observable as*

$$\langle \Phi | U^\dagger f U | \Phi \rangle = \sum_{j,k}^M c_{j,k} \prod_p^B \langle \phi^p | U_j^{p\dagger} f^p U_k^p | \phi^p \rangle, \tag{5.32}$$

where c is a matrix of coefficients and U_g^b is the g-th *unitary arising from the operator-Schmidt decomposition of all two-qubit gates acting on qubits in different sets, acting on partition b.*

Remark. *If every decomposed two-qubit gate is of the same kind, we have $M = n_s^N$, where n_s is the Schmidt number of the decomposed gate, and N the number of decomposed gates.*

The above proposition yields a method to simulate circuits which can be broken up into sub-circuits that are sparsely connected via cut two-qubit gates. This framework will be used to define a variational circuit model where the partitions are predefined and the placement of resourceful two-qubit gates is a design choice. We say these resourceful two-qubit gates creates non-local entanglement. Given access to classical memory, for B partitions we only have to compute and store $B \times n_s^N$ different states $U_k^p|\phi^p\rangle$, which can consequently be used to compute each inner product. If each partition contains n qubits, then the space complexity of the simulation method is $\mathcal{O}(n_s^N B 2^n)$[3]. By setting $n_s = 2$ for the CNOT gate we prove theorem 5.0.4.

Our final research question concerns testing the performance of quantum models with variable computational hardness in a QML setting. With the Circuit Partitioning formalism we can simulate circuits which contain parameterized single qubit gates, non-resourceful CNOTs which act on qubits within the same partition, and resourceful CNOTs, which act on qubits in different partitions. Since it is a design choice how many and where to place resourceful CNOTs, we have exact control over computational hardness of the circuit. For this reason we develop and use a simulator based on the Circuit Partitioning formalism for the next section of this work, and leave a numerical investigation with Matchgates as future work.

[3]For the simulation of quantum circuits it is typically the spacial complexity (memory) that is the limiting factor, particularly for constant depth circuits. For this reason we do not discuss the time complexity here.

Chapter 6

The power of VQC models in Machine Learning

In the previous section we developed three new algorithms to simulate Matchgate circuits containing SWAP gates, or Partitioned circuits. Whilst these algorithms might find use in intermediate-scale device verification, we aim to use them to investigate how the capacity and performance of parameterized versions of these circuits change with respect to their simulation difficulty. The results will give an indication of whether truly intrinsic quantum properties are necessary to achieve practical quantum advantages in a learning setting.

At present, Variational Quantum Circuits (VQCs) are the most common ML model implementable on NISQ devices for supervised, generative or reinforcement learning [5, 53]. They have been used in numerous classical ML settings, such as classification [43], generative tasks [54], learning temporal sequences [55] and image recognition [56] to mention just a few examples. We focus on classification and regression tasks.

Our final research question, which we tackle in this section, is to investigate how the computational hardness of a family of quantum models influences the performance of the model in supervised learning tasks. The family of models we use are a class of VQCs simulated using the Partitioned Circuit method. The two main factors governing the simulation difficulty of this class of circuits are the number of qubits per block, and the number of cut gates. By designing the model circuit based on a fixed number of qubits for some number of blocks, we have isolated the number of cut gates as the primary variable affecting the difficulty of the simulation. The aim of our experiments is to see how the performance changes as a function of the number of cut gates. By placing the cut gates randomly, we avoid introducing design bias and consequently artificially restricting the hypothesis class or influencing the results.

With our partitioned model the aim is to push the total number of qubits, given by $n_{tot} = B \times n$, beyond what is feasible with brute-force classical simulation. For most datasets we use $n = 5$, and determine B based on the number of features per dataset. Our key experiment to address the final research question is thus a comparison of the model performance against the number of cut gates. Before we can address this question there are a number of design choices which must be considered. These include the optimization algorithm, the model observable and the number of data-reuploading layers. We reduce the bias in our results by using multiple datasets. Our experimental settings are presented in Section 6.1, we justify open design choices with empirical observations reported in Section 6.2 in the lead-up to the results of our final experiment.

6.1 Experimental Methods

In this section we set out our Machine learning methodology. We first present the framework for our partitioned model, followed by details of the various optimization procedures tested in this work. Finally, in Section 6.1.3 we describe the datasets used in this report.

6.1.1 Model ansatz

We develop our model based on the Partitioned circuit formalism. The core building block of our model is one partitioned set of n qubits (called a "block"). On the pth block acts a variational circuit $U^p(\boldsymbol{\theta}^p, \mathbf{x}^p)$. This variational circuit performs both the qubit encoding of the data $\mathbf{x}$, and a parameterized unitary transformation defined by the set of parameters $\boldsymbol{\theta}^p$. The parameterization of the encoding gate increases the flexibility of the model, for convenience we absorb the parameters $\boldsymbol{\lambda}^p$ into $\boldsymbol{\theta}^p$. This variational circuit can be repeated L times in sequence with a unique set of parameters, to generate the notion of "layers" $U^p(\boldsymbol{\theta}^p, \mathbf{x}^p) \rightarrow \prod_{l=1}^{L} U_l^p(\boldsymbol{\theta}_l^p, \mathbf{x}^p)$ and yield a data re-uploading model. These variational circuits are interlaced with pre-cut gates which act to entangle different blocks. Finally, parameterized arbitrary single qubit rotations $V^p(\boldsymbol{\eta^p}) = \bigotimes_{q=1}^{n} U_{rot}(\boldsymbol{\eta}_q^p)$ are applied at the end of the circuit to allow the model to learn the optimal measurement basis. These rotations are implemented as $R_z(\alpha_j)R_y(\beta_j)R_z(\gamma_j)$, giving $\boldsymbol{\eta}_j = (\alpha_j, \beta_j, \gamma_j)^T$.

For our variational circuit we elect to go with a hardware efficient ansatz, as the gate set is native to many NISQ devices [57] and universal, meaning it can be used to approximate any unitary transformation to arbitrary precision [58]. A circuit diagrammatic representation is shown in Fig. 6.1.

The most important remaining design choices are the placement of cut gates (if any), and the observable. Whilst for our final experiment we place the cut gates randomly, it will be beneficial to define a placement method of our cut gates to increase the consistency across preliminary testing. We refer to models with some (no) cut gates as entangled (non-entangled) models. For our entangled model the method of placement is demonstrated in Fig. 6.2. Our cut gate of choice is the CNOT as it has a Schmidt number of two and hence offers the most favourable scaling. Since the intra-block CNOTs act before the inter-block CNOTs, the overall entanglement structure at a per-qubit level is not exactly cyclic, and only for $L > 1$ do we have full qubit connectivity. For our final experiment cut CNOTs can randomly entangle any pair of qubits in different blocks, but only act between variational layers up until the final layer of single qubit rotations. Our choice of observable is Pauli-Z. We empirically test the differences between a Pauli-Z on all qubits, on the first qubit of each block and on the first qubit in the first block below. The former two observables guarantee every data feature is part of the decision function defined by the model, whilst for the single qubit observable it depends on the inter-block entanglement and the number of blocks. Preliminary results showed that entangling structures

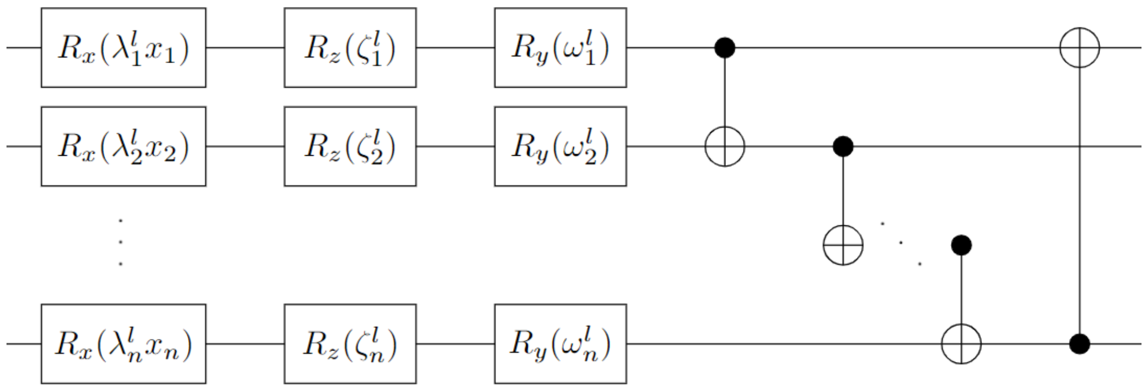


Figure 6.1: A circuit diagram of our variational circuit ansatz $U_l^p(\boldsymbol{\theta}^l, \mathbf{x}^p)$. Together the sequences $\boldsymbol{\lambda}^l, \boldsymbol{\zeta}^l$ and $\boldsymbol{\omega}^l$ form the set of parameters denoted by $\boldsymbol{\theta}^l$. Each block is assigned a subset of n feature vectors $\{x_i\}$, which are encoded via a parameterized $R_X(\lambda)$ single qubit rotations, such that each rotation angle is $\lambda_i x_i$. The data encoding gates are followed by R_Z and R_Y parameterized single qubit rotations as the core variational component of our model. This is followed by a cyclic chain of CNOT gates, entangling each qubit.

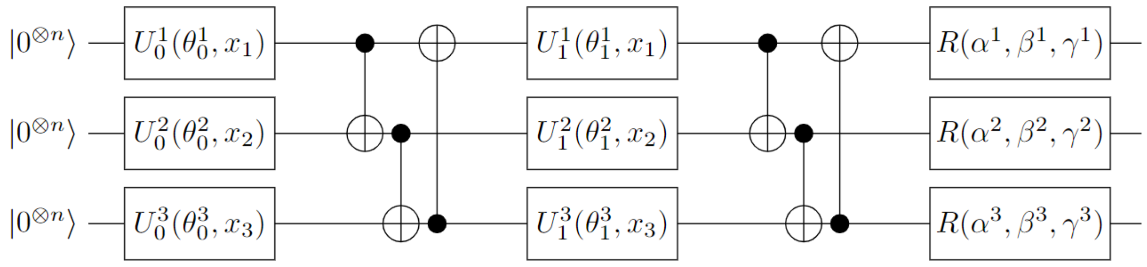


Figure 6.2: Circuit diagram of the Entangling model used in this report, with 2 layers and $N = 6$ cut CNOTs. Each U_l^j is one layer as shown in the blue box in Fig. 6.1. All CNOTs (acting across block partitions) have the last qubit in the control block as the control qubit, and the first qubit in the target block as the target qubit.

resulting in only a restricted set of features contributing to the decision function performed worse than non-restricting architectures.

Based on the above elements, we define our model hypothesis class;

Definition 6.1.1 (Partitioned circuit model)**.** The model labels a datapoint $\mathbf{x}$ by

$$F(\mathbf{x})_{\boldsymbol{\Theta}} = \langle \mathbf{0} | U^\dagger(\boldsymbol{\Theta}, \mathbf{x}) O U(\boldsymbol{\Theta}, \mathbf{x}) | \mathbf{0} \rangle,$$

which is computed using Proposition 5.2.1. Specifically we have

$$U_j^p = V(\boldsymbol{\eta}^p) \prod_l^L W_j^l U^l(\boldsymbol{\theta}_l^p, \mathbf{x}^p) \tag{6.1}$$

where W_j represents the gates arising from cut gates acting on partition p at layer l, V the single qubit rotations and U the data encoding and hardware efficient ansatz. We use the following fiducial state; $|\phi^p\rangle = |\mathbf{0}\rangle$. O is the Pauli-Z observable. For simplicity we have collected all parameters $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}$.

The above model outputs a label within the range defined by the eigenvalues of the observable, which for our case is [-1,1]. For classification tasks whereby we require the output to be a probability we rescale this to the range [0,1], which is equivalent to replacing the Pauli-Z observable with an all-zero projector $|\mathbf{0}\rangle\langle\mathbf{0}|$ on the measured qubits.

6.1.2 Optimization procedure & Trainability

Many ML models with free parameters suffer from exploding or vanishing gradients during training with gradient-based optimizers, especially if the parameters are randomly initialized [59]. This phenomenon describes the situation where the gradients, and hence the value by which parameters should change in order to move to an optimal solution, become exponentially large or small with respect to some parameter related to the model complexity. VQC models were first shown to suffer from vanishing gradients, commonly dubbed "Barren Plateaus", by McClean *et al.* [60], and the issue remains an active field of research. The first evidence for barren plateaus by McClean *et al.* considers randomly initialized Hardware Efficient Ansatze, which are our variational circuit of choice. They show such circuits correspond to a unitary 2-design, ie they match the Haar distribution up to the second moment [61], which can be used to prove the variance of the gradient of the loss function decreases exponentially with the number of qubits. As we are using this ansatz we must be vigilant that Barren plateaus might affect the training of our models, especially for larger models.

This does not preclude using gradients for training. In our model our parameterized gates satisfy the requirements to analytically calculate the gradient using the parameter shift rule [62], which only requires two further circuit evaluations. As each parameter only appears once in one gate, when we apply the chain rule to the product of partitions only one term is non-zero. The gradient still has to be calculated individually for each term in the sum in Eq. (5.32), such that $2M$ circuits have to be evaluated overall.

To determine the best optimization method we performed some preliminary testing of the Adam, Covariance matrix adaptation evolution strategy (CMA-ES), Simultaneous perturbation stochastic approximation (SPSA), L-BFGS and Coordinate-wise descent optimization routines. We find the gradient-based Adam optimizer to be the best in terms of obtained loss, stability, rate of convergence and processing time for models with 2 blocks of 5 qubits each. For this reason we primarily use the Adam optimizer in our experiments.

6.1.3 Datasets

We repeat our experiments on multiple datasets to minimize the impact of the underlying decision functions of the datasets on our results. To achieve this we pick a number of classical and quantum datasets. Classical data covers samples of images, text and real world macroscopic observations (e.g. temperature readings), whilst quantum data is generated from quantum process. All the chosen real-world datasets originate from the UCI repository. A brief description of each dataset is given below, and a summary is given in Table 6.2.

WDBC [63]
The Wisconsin Breast Cancer (Diagnostic) dataset considers various geometric features of cell nuclei computed from medical images of breast masses. The task is to identify the sample as either malignant or benign, which is a binary classification problem. In total there are 10 such geometric features, and our only preprocessing of the features is to rescale them to the range $[-\pi/2, \pi/2]$.

ION [64]
The Ionosphere dataset concerns the identification of structure amongst free electrons in ionosphere. The raw data was collected by a radar system, which had 17 pulse numbers. This was processed by an autocorrelation function which yields 17 complex values, one per pulse number. The dataset is obtained by splitting the real and complex parts into their own features, resulting in 34 features in total. It is a binary classification task. One feature has zero variance so is dropped. Furthermore some pairs of features show a high degree of correlation, so we use Principle Component Analysis (PCA) to allow us to obtain a variable number of features (we always use the first N principle components when asked for N features). The output principle components are rescaled to the range $[-\pi/2, \pi/2]$.

SPECTF [65]
For this dataset Single Proton Emission Computed Tomography (SPECT) has been used to obtain images to assess the cardiac health of a patient. The images were processed to give 44 features, and labelled as either normal or abnormal. The features exhibit a strong degree of correlation amongst themselves so we use PCA to obtain a more efficient representation. Again the resulting features are rescaled to the range $[-\pi/2, \pi/2]$.

MNIST [66]
We use an alternative handwritten digit dataset to the popular benchmarks in Machine Learning. Specifically the 32x32 bitmaps are segmented into 4x4 non-overlapping blocks. Within each block the number of "on" pixels are counted and returned, resulting in an 8x8 matrix with integers in the range $[0, 16]$. We rescale these integers to the range $[0, \pi]$, as this is a more appropriate range for our data encoding strategy. The dataset contains 10 unique classes corresponding to the digits 0 to 9. We restrict the dataset to samples only from the classes corresponding to the digits 2 and 3, resulting in a binary classification task.

VQC output
Our artificial dataset is the output of our partitioned model initialized with random weights sampled from the uniform distribution on $[-\pi/2, \pi/2]$. The underlying decision function is hence quantum in nature. We generate 1000 input vectors from a uniform distribution over $[-\pi, \pi]$, and the label is given by the output of the model (the expectation value of a Pauli Z observable on the first qubit in each block). The model parameters are given in Table 6.1.

Blocks	Qubits per block	Layers	Inter-block entanglement
2	5	2	$q_{1,5}$ on $q_{2,1}$, $q_{2,5}$ on $q_{1,1}$

Table 6.1: Model parameters used to generate the VQC output dataset. $q_{i,j}$ on $q_{k,l}$ represents a CNOT where the jth qubit in the ith block is the control qubit and the lth qubit in block k is the target qubit. The inter-block CNOTs only acts in between the first and second layer.

Ising dynamics
For our second quantum dataset we take the critical transverse field Ising Hamiltonian $H = \sum_j^{N-1} Z_j Z_{j+1} + \sum_i^N X_i$ to generate a regression task. We simulate the Hamiltonian using the Lie/Trotter product formula $exp(-i \sum_j^m H_j t) = (\prod_j^m exp(-iH_j t/\tau))^\tau$. The data is obtained from a 10 qubit system, with $t/\tau = 0.08$ and an all-zero fiducial state. The data feature is time, and the output is a Pauli Z observable on the first qubit. We obtain the output as follows: The simulation is run for 3750 timesteps (ie for t=300); for each subsequent timestep, we evolve the wavefunction as before, and record the expectation value $\langle Z \rangle$ on the first qubit. This is repeated for 100 timesteps (ie from t=300 to t=308) to generate an equivalent number of datapoints. The time span is mapped from the range [300, 308] to [-π, π] to give our final data attribute.

Name	Type	Task	# of features	Preprocessing
CANCER	Medical	Classification	10	Rescaling
ION	RWO	Classification	33	PCA & rescaling
SPECTF	Medical	Classification	44	PCA & rescaling
MNIST	Images	Classification	64	Rescaling
VQC output	Quantum	Regression	10	None
ISING	Quantum	Regression	1	Rescaling

Table 6.2: Summary of the main features of each dataset. RWO is an acronym for Real World Observation.

6.2 Numerical Results

In this work we have developed two new algorithms for computing the expectation value of matchgate circuits supplemented with SWAP gates, and one for arbitrary circuits with little non-local entanglement. In this section we implement the latter to demonstrate the relation between the degree of non-local entanglement of a circuit model and its performance in a learning setting. First we investigate the performance differences between the three proposed observables: Pauli Z on all qubits, on the first qubit in each block and on the first qubit in the first block. The mean validation accuracy is plotted in Fig. 6.3 for the first three real-world datasets. We define the validation accuracy as the accuracy obtained by the model on a test set (i.e. datapoints not used to train the model). Similarly, the validation loss is the loss obtained by the model on the test set. The validation accuracy is an important indicator of model performance in classification tasks, while the validation loss holds the same status in regression tasks.

It is evident the differences in validation accuracy between the three observables are smaller than the error bars. Notably we do notice a minor improvement in model performance for the non-entangled model for the ION dataset, compared to that of the entangled model. The non-entangled model with the "Single" observable performs as well as all the other models, which is interesting because only half of the data features contribute to the decision function. For the WDBC dataset we attribute this to the high discriminative power of a single feature already, whilst for

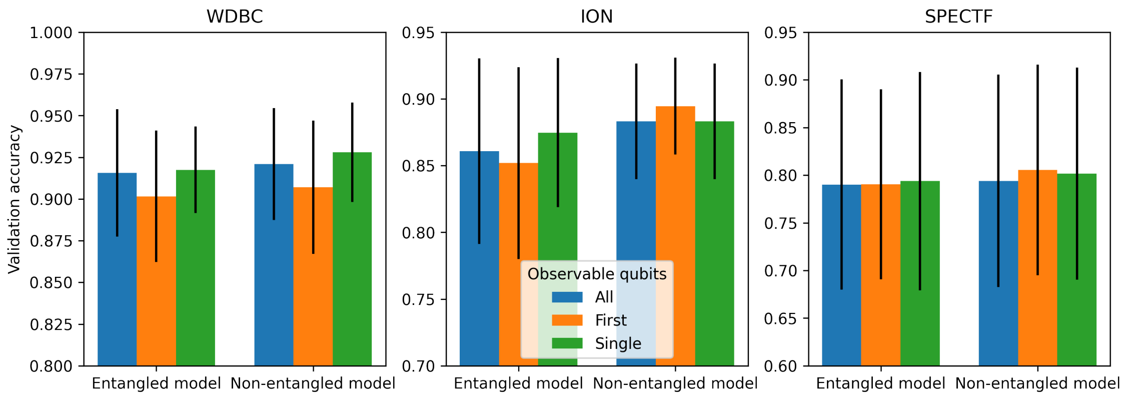


Figure 6.3: **Learning performance on real-world datasets for different observables.** The validation accuracy is computed as the accuracy on the entire test set after training, and the mean and first standard deviation is over 10-fold cross validation. Every model consisted of 2 blocks of 5 qubits with 4 layers.

the ION and SPECTF we used PCA preprocessing and used the first 10 features. Despite shuffling the order of each feature, this likely also decreased the difficulty of classifying the data. The above experiment was also performed with the same models but instead 1, 2 and 3 layers. More layers increases the performance, but it is still not possible to establish a superior choice of observable. It should be noted the results across all settings are competitive with classical models, based on results reported by the UCI repository they were obtained from.

We repeat the above study on the VQC dataset; the results are presented in Table 6.3.

Observable	**Training loss** ($\times 10^{-3}$)	**Validation loss** ($\times 10^{-3}$)	**Generalization loss** ($\times 10^{-3}$)
First	1.018 ± 0.023	1.293 ± 0.311	0.275 ± 0.318
All	1.014 ± 0.032	1.345 ± 0.294	0.331 ± 0.315
Single	1.038 ± 0.022	1.273 ± 0.295	0.235 ± 0.305
Neural Network	1.079 ± 0.100	1.583 ± 0.247	0.503 ± 0.299

Table 6.3: **Learning performance on the VQC output dataset for different observables.** The values in the table are the MSE loss. The same methodology is applied as in Fig. 6.3 to obtain a mean and standard deviation. Additionally we test a Neural Network with two layers of 50 neurons for comparison. It is likely the performance can be improved by architectural tuning and/or hyperparameter optimization.

The Neural Network converges to a significantly higher training and validation loss than the quantum models, providing good evidence that quantum models are better suited to learning quantum decision functions. All models exhibit overfitting as the training loss is much lower than the validation loss and is further evidenced by the much larger error bars on the validation loss. The All-qubit observable has the largest generalization loss, which we define as the validation loss minus the training loss, whilst the single qubit observable has the smallest. This can in part be understood by the difference in model complexities as for the all-qubit observable a larger number of parameters and gates contribute to the decision function relative to the single qubit observable. Repeating this study for a dataset generated using an all-qubit Pauli Z observable reinforces the findings. Strikingly, the variance of the labels for the VQC dataset is 1.095×10^{-3}, which is very similar to the training loss. One might hypothesize that the model is only learning to output the same distribution of the data generating model ie $\tilde{h}(x_i) = y_j \sim S_Y$, rather than learning the exact decision function $h(x_i) = y_i$. To test this hypothesis we repeat the experiment but with randomized labels, and report the results in Table 6.4.

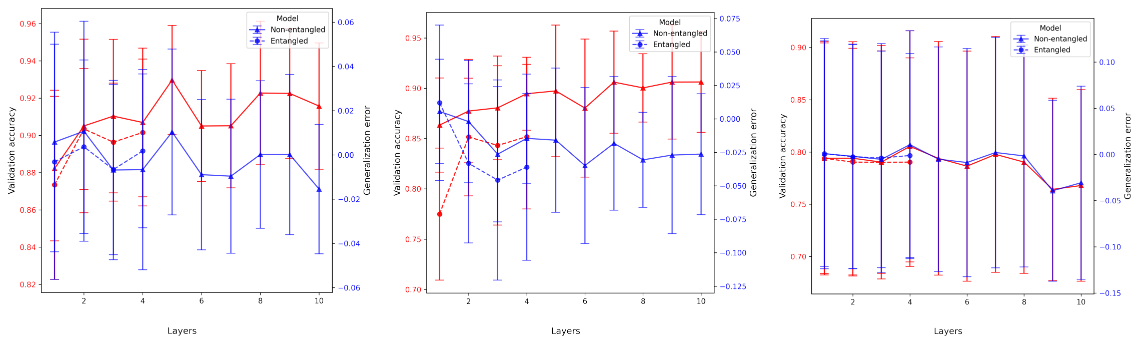
Observable	**Training loss** ($\times 10^{-3}$)	**Validation loss**($\times 10^{-3}$)	**Generalization loss**($\times 10^{-3}$)
First	1.425 ± 0.094	1.480 ± 0.230	0.055 ± 0.259
All	1.428 ± 0.080	1.493 ± 0.242	0.064 ± 0.264
Single	1.414 ± 0.083	1.453 ± 0.229	0.039 ± 0.262
Neural Network	1.244 ± 0.081	1.617 ± 0.274	0.373 ± 0.305

Table 6.4: **Learning performance on a quantum dataset with randomized labels for different observables.** The labels of the entire dataset were randomly shuffled before using 10-fold cross validation to obtain a mean and standard deviation.

By randomizing the labels we can distinguish between models that memorize the training dataset and those that learn the distribution of the training labels. Those that merely memorize the training set will generalize poorly as there is no correlation between the input and the labels. On the other hand, models that learn characteristics of the underlying label distribution P_Y will have a very small generalization error or loss as both the training and test sets are sampled from this distribution. From Table 6.4 it is evident the Neural Network maintains a large generalization loss, suggesting it is memorizing the dataset. On the other hand the quantum models are capable of learning P_Y as the generalization error is small. Key here is that the mean of P_Y is zero, meaning the best the model can do is to output zero for every input. This strategy would give a loss of 1.406×10^{-3}, which is slightly lower than the losses posted in Table 6.4. Hence we can conclude our quantum model is capable of learning P_Y when given randomized training data. Furthermore, since the losses on the original dataset are lower than $1,406 \times 10^{-3}$, we confirm our model is learning the underlying decision function, and disprove the hypothesis that the model is learning the label distribution P_Y only.

Comparing the different observables in Tables 6.3 and 6.4 the generalization loss suggests restricting the number of measured qubits acts to regularize the model. For this reason our observable of choice for the remainder of this investigation is the "First" observable: a product of Pauli Z's on the first qubit of each block. In particular we thereby guarantee that even in the absence of inter-block entanglement every data feature contributes to the decision function.

The last model parameter to optimize for the final experiment is the number of layers. We study the performance of the entangled and non-entangled models as a



(a) WDBC (b) ION (c) SPECTF

Figure 6.4: **Learning performance on real-world datasets for increasing numbers of layers.** The validation (red) and generalization (blue) loss for a range of layers for the 3 real-world datasets. Due to hardware limitations we are only able simulate up to 4 layers for the entangled model (corresponding to 8 cut gates). For each setting we perform 10-fold cross validation and report the mean and standard deviation.

function of the number of layers in Fig. 6.4 for the real-world datasets.

By the theoretical work of [15] the additional data-encoding gates introduced with an increasing number of layers should increase the capacity of the model, as well as through the parameterized single qubit rotations and entangling gates. The downside of adding more layers is the additional parameters increase the size of the parameter space that must be searched for the optimum solution, as well as increasing the run-time due to the additional matrix multiplications that must be computed. Based on the results in Fig. 6.4 we set the number of layers to 2 and 4 for the WDBC and ION datasets respectively. As the SPECTF dataset shows remarkable apathy to any set of model parameters and architectures we test it on, its use is discontinued for our remaining experiments. As the VQC dataset was generated with a model with 2 layers, we use the same number of layers for the final experiment.

Our final dataset, based on an Ising model Hamiltonian, was generated from a 10 qubit system. A very similar learning task using an equivalent amount of qubits was done by [23], who observed they were able to learn the dynamics using only a 6-qubit circuit. To verify this we attempt to learn the ISING dataset with the entangled and non-entangled models, both consisting of 2 blocks and 2 layers. We vary the number of qubits per block (qpb) from 2 through 7. The results are plotted in Fig. 6.5.

For the entangled model, the best mean validation loss is for the 5 qpb model, whilst the best training and validation loss occurs at 4 qpb. The mean losses for the non-entangled models decrease steadily up to 5qpb, after which it plateaus. The variance in the dataset labels is 0.055, which is significantly higher than the losses obtained in this experiment, which suggests the models are successfully learning. In the final experiment we train a 2 block model with 5 qubits per block for the ISING dataset. To assess the models ability to learn the actual evolution of system we evaluate them on a holdout set consisting of the 40 timesteps after the main dataset. The results are plotted in Fig. 6.6.

From Fig. 6.6 it is evident the quantum model is more than capable of learning

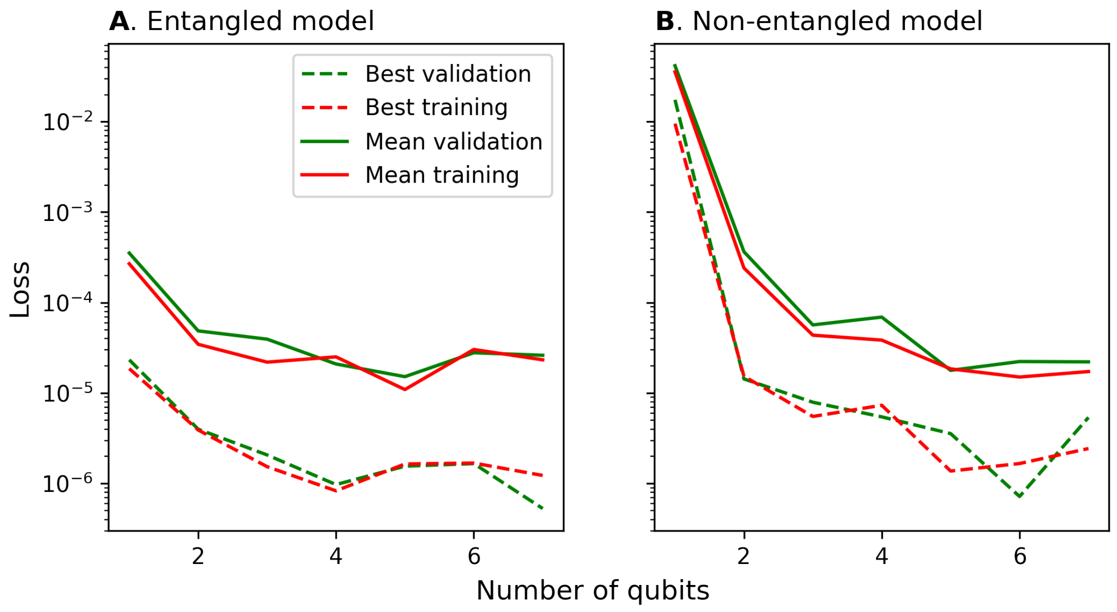


Figure 6.5: **Model performance on the ISING dataset for a varying number of qubits per block.** The mean was obtained over the results from 10-fold cross-validation, and is plotted with a solid line. The final training and validation loss varied over a few orders of magnitude so we find the error bars to be too large to plot. The loss obtained by the best performing model is plotted with a dashed line. Green lines indicate the validation loss, whilst the red lines are plots of the training loss.

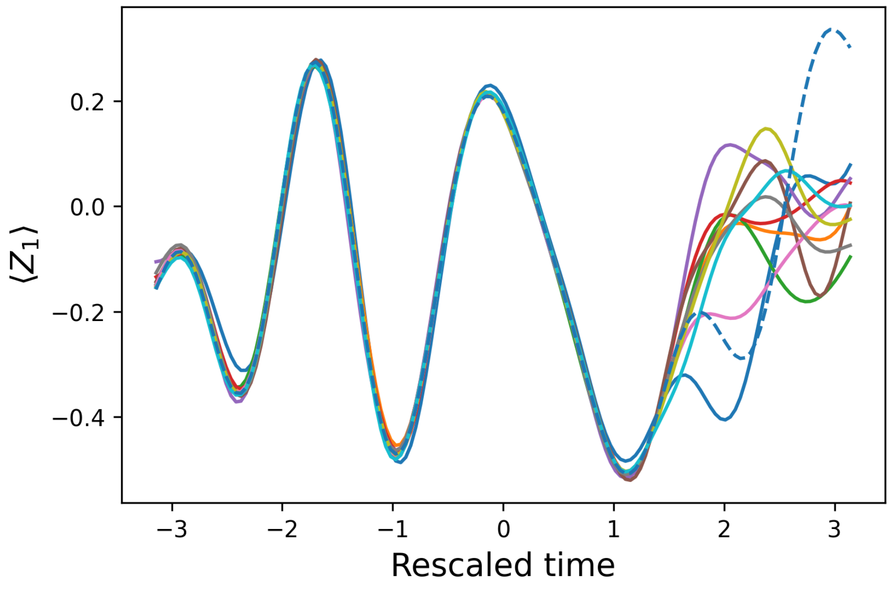
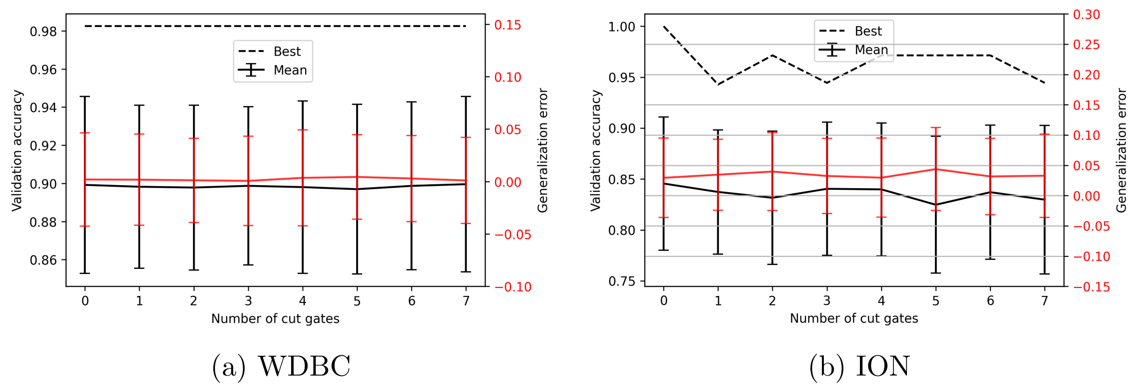
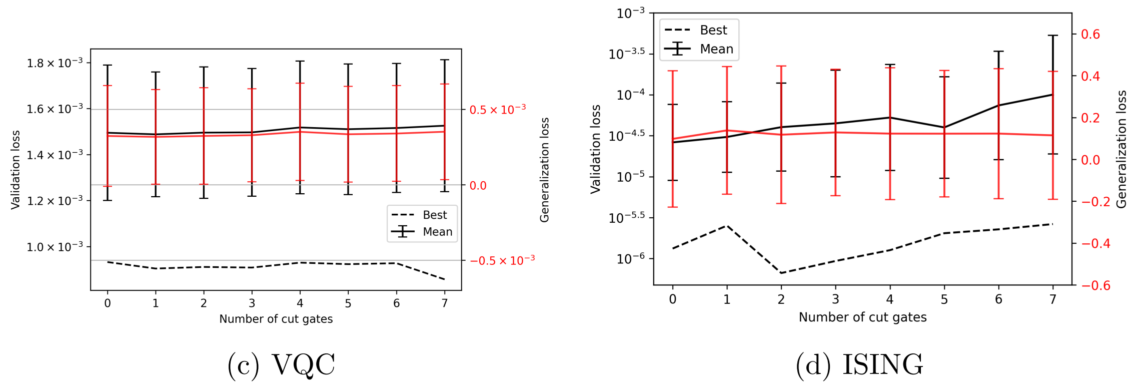


Figure 6.6: **Learning the evolution of an transverse Ising Hamiltonian.** The dashed line represents the true evolution, whilst the solid lines represent the functions learnt by 10 different models. The transition from the training and test set to the holdout set occurs at a rescaled time of 1.35.



(a) WDBC (b) ION



(c) VQC (d) ISING

Figure 6.7: **Learning performance for models with an increasing number of cut gates.** Both the mean and best validation losses are plotted on the left hand axis, for 16 runs of 10-fold cross-validation, giving 160 datapoints overall. On the right hand axis the mean generalization error (loss) is shown, given by $1/N \sum_i (f^i_{val} - f^i_{train})$ where f_{val} and f_{train} are the accuracies (losses) obtained on the entire validation and training set respectively, and we sum over all runs. An exception is the Ising dataset (d); here we instead take $1/N \sum_i (\log_{10}(L^i_{val}) - \log_{10}(L^i_{train}))$.

the function representing a system governed by a transverse Ising Hamiltonian. However, it is not capable of learning and reproducing the system dynamics as the predictions for the holdout poorly represent the true evolution.

So far we have not presented any results for the MNIST dataset. The goal was to train a model of 8 blocks of 8 qubits, which would represent one of the largest QML models to be reported in literature. Unfortunately it has not been possible to train the model using the Adam or SPSA routines for a large range of hyperparameter settings. The CMA, LBFGS and Coordinate descent routines are prohibitively slow to run on the hardware available. It is likely some heuristic training algorithm is necessary to circumvent the issue of exponentially degrading gradients.

We now have all the experimental settings to evaluate the learning performance of our quantum model as a function of the degree of non-local entanglement only, which we measure by the number of cut gates. The results are plotted in Fig. 6.7.

For both the WDBC, ION and VQC datasets, it is clear the number of cut gates has no observable effect on the validation accuracy or generalization error. The error bars cover a range of around 5, 10 and 20 percent respectively. As the range of the error bars also does not change with respect to the number of cut gates, the variability likely arises due to the intrinsic trainability of the model. A comparison of the cut gate placement for models performing within, above and below the error bars does not unveil particular placements which are especially beneficial or detrimental to the performance. With the ISING dataset we actually observe the addition of cut gates leads to a decrease in performance. Across all datasets the generalization error or loss is independent of the number of cut gates, which suggests the self-regularizing aspect of our quantum model is not due to the difficulty of quantum simulation, but rather due to some quantum characteristic such as the unitarity of evolution [23]. Alternative explanations have also been put forward by eg [15] who tie the regularization to the depth of the data re-uploading circuit.

6.3 Discussion

The goal of QML is to utilize quantum information theory to create better machine learning methods. The notion of "better" is purposely left ambiguous as there are many ways to compare quantum and classical methods. Here we focus simply on learning performance of a VQA. VQAs are a NISQ-friendly, flexible approach to QML as they utilize both classical and quantum processing units. Classical data can be converted into a quantum state via qubit encoding or the more difficult to implement amplitude encoding. By parameterizing a quantum system, such as the gates in a circuit, and choosing an observable one has created a hypothesis class which can be trained in a classical optimization loop.

In this work we have used a VQA to explore the relation between the computational hardness of a quantum model and it's machine learning performance. Our circuit partitioning scheme allows us to systematically control the computational hardness of the circuit through the dosed inclusion of non-local entanglement via cross-partition CNOTs. The computational hardness of this scheme is primarily dictated by the spacial complexity. We investigated research question 3 with a partitioned circuit model based on a data re-uploading Hardware Efficient Ansatz and a selection of classical and quantum supervised learning tasks.

First we empirically justified some model features that do not affect the spacial complexity and hence computational hardness. These include the choice of observable, and the number of variational layers (and thereby also the number of data re-uploading gates). In these studies we found both our entangled and non-entangled models were already capable of competing with common classical methods on real-world datasets, giving an early indication the non-local entanglement is not an important factor in model performance. Learning the output of VQC was a more difficult task, despite using the exact same circuit to generate the data as to learn. This might be due to the fact we chose to use the expectation value as our label, rather than the state vector output by the data generating circuit. Since each possible value of the expectation value does not have one unique quantum state associated to it (under reasonable conditions), the loss function will have a local minimum when the model outputs any of the quantum states corresponding to that expectation value. Consequently, gradient based methods are highly likely to get stuck in these local minima and fail to converge to the global minimum. We do find our model is very capable of learning the function defined by the evolution of a system described by a transverse Ising Hamiltonian.

In our final experiment to address research question 3, the cut gates are placed randomly. We conclude the performance is independent of the number of cut gates for the real-world datasets. Our general methodology was to design the model as to maximize the performance of both entangled and non-entangled variants of our model. As we were already able to achieve competitive results in these preliminary investigations (eg Fig. 6.3) it is possible we hit a natural ceiling in the separability of the data within the hypothesis class. The quantum dataset focused on learning the evolution of a transverse Ising Hamiltonian was more difficult for the model to learn as the number of cut gates increased. This might be due to there being only one feature, namely time. The additional entanglement creates higher order terms in the model function that are not necessarily present in the underlying decision function.

It should be noted that the number of possible configurations in which cut gates can be placed increases combinatorially in the number of cut gates. As we tested around 160 configurations for each number of cut gates, this is only a small fraction of the possible placements for the models with more than one cut gate, decreasing the probability of finding the optimal placements and consequently of finding the true best possible model performance. By keeping the number of layers low we restrict the space of possible configurations, but this also restricts the performance benefit anticipated for data re-uploading models such as the one used in this work.

Our results provide further evidence that potent quantum learning models can be created by combining multiple smaller circuits. This allows practitioners to use multiple devices to create models with qubit counts beyond what is available with state of the art devices. Furthermore, these smaller circuits can already be designed and optimized to achieve competitive performances without needing to entangle the circuits. Due to vanishing gradients training does get more difficult as the model size increases, however due to the disjoint sets of parameters, clever training methods may circumvent this issue whereby one uses both the gradients of single blocks as well as the total gradient. Non-entangled models may even be trained individually when using projectors as an observable. Such non-entangled structures have strong parallels with ensemble learning or multiple classifier systems, which are well established classical techniques in Machine Learning. A study of these parallels may be an interesting avenue for further research.

Chapter 7

Conclusions

In this thesis we have studied the features that determine the computational hardness of classically simulating quantum systems and circuits. Moreover, we investigated the impact one example of such a feature has on the performance of a quantum model in a supervised machine learning setting. Studying such features is important to better understand how to maximally utilize quantum devices to compute systems larger than they are themselves. It can also be used to guide expectations surrounding the future of QML.

In our search of finding simulation methods whereby we can systematically control the computational hardness, we find the interaction and entanglement of distant qubits to be an important feature in both the Matchgate formalism and the circuit model of quantum computing. This influences the extent to which they can be simulated by classical algorithms. Within the matchgate formalism these interactions are realized by the SWAP gate, which is not a matchgate but still preserves parity, a key symmetry in matchgates. For arbitrary quantum circuits the computational hardness is in part determined by the degree to which the circuit can be partitioned into subcircuits by cutting two-qubit gates.

By developing algorithms around these specific features we obtain exact simulation methods which scale polynomially in the system size but exponentially in the number of SWAP or cut gates. We develop two new algorithms for the simulation of Match- + SWAP gate circuits, one of which achieves the semi-strong level of simulation and scales as $\mathcal{O}(4^{2N}\text{poly}(n, N)$. This is the same algorithmic scaling as the SWAP gadget approach, which belongs to the weak level of simulation. As such our algorithm represents the current state of the art method for Match- + SWAP gate simulation. Furthermore we develop the notion of circuit partitioning, based on the gate decomposition technique, and develop code for the simulation of a class of partitioned circuits. It allows for the linear scaling in the total number of qubits by keeping the partitions a constant size, whilst the scaling is exponential in the number of cross-partition entangling gates.

Finally, we use the simulator to investigate the learning capabilities of a VQC and the influence of the simulation difficulty on it's performance. Our model ansatz is competitive with classical methods for a range of real-world tasks, and outperforms a small selection of Neural networks when learning the output of a quantum circuit. Most significantly we find the presence of cross-partition gates to have no effect on the learning performance, and even be detrimental when learning the dynamics of a quantum Ising system.

The results of this thesis have raised further questions, both on matters regarding computational hardness as well as learning capabilities. Our work on Matchgates has focused on exact methods only: there is a notable under-representation of approximate matchgate simulation methods in literature, whilst this could be a worthwhile direction to explore. Furthermore, we have explored the SWAP gate only. It remains an open question if there exists a decomposition of a non-matchgate parity preserving two-qubit unitary into matchgates that has fewer than four terms. The basis defined by the Pauli products (plus the I$\otimes$Z and Z$\otimes$I matchgates) is insufficient to achieve this, as any sum of two or three of these terms is either a matchgate or not unitary. This can be proven by evaluating all $C^6_2 + C^6_3 = 35$ possible combinations. Similarly, a SWAP gadget which requires fewer adaptive measurements would also improve it's scaling, allowing it to compete with our SWAP decomposition for problems where sampling the output of the circuit is sufficient.

One of our research question was to investigate the learning performance as a function of the simulation difficulty of quantum models. This is a very broad question and our findings are limited by our choice of algorithm and the associated quantum feature which dominates the simulation difficulty. Evidently, our measure of non-local entanglement is not an important factor of learning performance. Are there other schemes whereby a different quantum feature is isolated and simulated such that it is a significant indicator of learning capability? Despite the limited scope of our numerical results, this thesis provides a basic recipe to systematically identify which aspects of quantum models enable them to learn.

Bibliography

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, 10 2009.

[2] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 124–134, 1994.

[3] Seth Lloyd. Universal Quantum Simulators. *Science*, 273(5278):1073–1078, 8 1996.

[4] Jonathan Allcock and Shengyu Zhang. Quantum machine learning. *National Science Review*, 6(1):26–28, 1 2019.

[5] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics 2021 3:9*, 3(9):625–644, 8 2021.

[6] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2 2020.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 9 2017.

[8] Scott Aaronson. Read the fine print. *Nature Physics 2015 11:4*, 11(4):291–293, 4 2015.

[9] Martin Plesch and Časlav Brukner. Quantum-state preparation with universal gate decompositions. *Physical Review A - Atomic, Molecular, and Optical Physics*, 83(3):032302, 3 2011.

[10] Maria Schuld and Nathan Killoran. Is quantum advantage the right goal for quantum machine learning? 3 2022.

[11] Daniel Gottesman. The Heisenberg Representation of Quantum Computers. 7 1998.

[12] Barbara M. Terhal and David P. DiVincenzo. Classical simulation of noninteracting-fermion quantum circuits. *Physical Review A*, 65(3):032325, 3 2002.

[13] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics 2019 1:9*, 1(9):538–550, 8 2019.

[14] Richard Jozsa and Akimasa Miyake. Matchgates and classical simulation of quantum circuits. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 464(2100):3089–3106, 12 2008.

[15] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 3 2021.

[16] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, 2010.

[17] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 9 1989.

[18] Andrew Chi Chih Yao. Quantum circuit complexity. *Annual Symposium on Foundatons of Computer Science (Proceedings)*, pages 352–361, 1993.

[19] Scott Aaronson and Lijie Chen. Complexity-Theoretic Foundations of Quantum Supremacy Experiments. 2016.

[20] Richard Jozsa and Maarten van den Nest. Classical simulation complexity of extended Clifford circuits. *Quantum Information and Computation*, 14(7-8):633–648, 5 2013.

[21] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 11 2019.

[22] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3), 4 2018.

[23] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 9 2018.

[24] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature 2019 567:7747*, 567(7747):209–212, 3 2019.

[25] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, 6(2):021043, 6 2016.

[26] https://github.com/MariusvanLaar/MSc-thesis.

[27] Sergey Bravyi, David Gosset, and Ramis Movassagh. Classical algorithms for quantum mean values. *Nature Physics 2021 17:3*, 17(3):337–341, 1 2021.

[28] L. G. Valiant. Quantum computers that can be simulated classically in polynomial time. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 114–123, 2001.

[29] Daniel J. Brod and Ernesto F. Galvão. Extending matchgates into universal quantum computation. *Physical Review A - Atomic, Molecular, and Optical Physics*, 84(2):022310, 8 2011.

[30] M. Hebenstreit, R. Jozsa, B. Kraus, and S. Strelchuk. Computational power of matchgates with supplementary resources. *Physical Review A*, 102(5):052604, 11 2020.

[31] Richard Jozsa, Barbara Kraus, Akimasa Miyake, and John Watrous. Matchgate and space-bounded quantum computations are equivalent. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 466(2115):809–830, 3 2010.

[32] Wei Tang, Teague Tomesh, Martin Suchara msuchara, Jeffrey Larson, and Margaret Martonosi. CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations. *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021.

[33] Michael A. Perlin, Zain H. Saleem, Martin Suchara, and James C. Osborn. Quantum Circuit Cutting with Maximum Likelihood Tomography. *npj Quantum Information*, 7(1), 5 2020.

[34] Jacob Biamonte and Ville Bergholm. Tensor Networks in a Nutshell. 7 2017.

[35] Xiao Yuan, Jinzhao Sun, Junyu Liu, Qi Zhao, and You Zhou. Quantum Simulation with Hybrid Tensor Networks. *Physical Review Letters*, 127(4):040501, 7 2021.

[36] F. Barratt, James Dborin, Matthias Bal, Vid Stojevic, Frank Pollmann, and A. G. Green. Parallel quantum simulation of large systems on small NISQ computers. *npj Quantum Information 2021 7:1*, 7(1):1–7, 5 2021.

[37] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. Simulating Large Quantum Circuits on a Small Quantum Computer. *Physical Review Letters*, 125(15):150504, 10 2020.

[38] Keisuke Fujii, Kaoru Mizuta, Hiroshi Ueda, Kosuke Mitarai, Wataru Mizukami, and Yuya O. Nakagawa. Deep Variational Quantum Eigensolver: A Divide-And-Conquer Method for Solving a Larger Problem with Smaller Size Quantum Computers. *PRX Quantum*, 3(1):010346, 3 2022.

[39] Simon C. Marshall, Casper Gyurik, and Vedran Dunjko. High Dimensional Quantum Machine Learning With Small Quantum Computers. 3 2022.

[40] Diego Ristè, Marcus P. Da Silva, Colm A. Ryan, Andrew W. Cross, Antonio D. Córcoles, John A. Smolin, Jay M. Gambetta, Jerry M. Chow, and Blake R. Johnson. Demonstration of quantum advantage in machine learning. *npj Quantum Information 2017 3:1*, 3(1):1–5, 4 2017.

[41] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics 2021 17:9*, 17(9):1013–1017, 7 2021.

[42] Maria Schuld. Supervised quantum machine learning models are kernel methods. 1 2021.

[43] Amira Abbas, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science 2021 1:6*, 1(6):403–409, 6 2021.

[44] Matthias C. Caro, Elies Gil-Fuster, Johannes Jakob Meyer, Jens Eisert, and Ryan Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum*, 5:582, 11 2021.

[45] Casper Gyurik, Dyon van Vreumingen, and Vedran Dunjko. Structural risk minimization for quantum linear classifiers. 5 2021.

[46] Evan Peters, João Caldeira, Alan Ho, Stefan Leichenauer, Masoud Mohseni, Hartmut Neven, Panagiotis Spentzouris, Doug Strain, and Gabriel N. Perdue. Machine learning of high dimensional data on a noisy quantum processor. *npj Quantum Information*, 7(1), 1 2021.

[47] Tobias Haug, Chris N. Self, and M. S. Kim. Large-scale quantum machine learning. 8 2021.

[48] M. Hebenstreit, R. Jozsa, B. Kraus, S. Strelchuk, and M. Yoganathan. All Pure Fermionic Non-Gaussian States Are Magic States for Matchgate Computations. *Physical Review Letters*, 123(8):080503, 8 2019.

[49] Daniel J. Brod. Efficient classical simulation of matchgate circuits with generalized inputs and measurements. *Physical Review A*, 93(6):062332, 6 2016.

[50] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A - Atomic, Molecular, and Optical Physics*, 71(2):022316, 2 2005.

[51] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man Hong Yung, Xiao Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications 2014 5:1*, 5(1):1–7, 7 2014.

[52] Michael A. Nielsen, Christopher M. Dawson, Jennifer L. Dodd, Alexei Gilchrist, Duncan Mortimer, Tobias J. Osborne, Michael J. Bremner, Aram W. Harrow, and Andrew Hines. Quantum dynamics as a physical resource. *Physical Review A*, 67(5):052301, 5 2003.

[53] Sofiene Jerbi, Casper Gyurik, Simon C. Marshall, Hans J. Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning. 3 2021.

[54] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Variational quantum Boltzmann machines. *Quantum Machine Intelligence*, 3(1):1–15, 6 2021.

[55] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L. Fang. Quantum Long Short-Term Memory. 9 2020.

[56] Junhua Liu, Kwan Hui Lim, Kristin L. Wood, Wei Huang, Chu Guo, and He Liang Huang. Hybrid quantum-classical convolutional neural networks. *Science China Physics, Mechanics & Astronomy 2021 64:9*, 64(9):1–8, 8 2021.

[57] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature 2017 549:7671*, 549(7671):242–246, 9 2017.

[58] Jean Luc Brylinski and Ranee Brylinski. Universal quantum gates. *Mathematics of Quantum Computation*, pages 101–116, 1 2002.

[59] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19, 2006.

[60] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications 2018 9:1*, 9(1):1–6, 11 2018.

[61] Aram W. Harrow and Richard A. Low. Random Quantum Circuits are Approximate 2-designs. *Communications in Mathematical Physics 2009 291:1*, 291(1):257–302, 7 2009.

[62] Gavin E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. 5 2019.

[63] Street W Mangasarian Olvi Wolberg William. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995.

[64] Wing S Hutton L Baker K Sigillito V. Ionosphere. UCI Machine Learning Repository, 1989.

[65] Kurgan Lukasz Goodenday Lucy Cios Krzysztof. SPECTF Heart. UCI Machine Learning Repository, 2001.

[66] C Alpaydin E. & Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998.