Separating quantum and classical computing: rigorous proof and practical application

Marshall, S.C.

Separating quantum and classical computing: rigorous proof and practical application

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl
volgens besluit van het college voor promoties
te verdedigen op dinsdag 27 mei 2025
klokke 10 uur

door

Simon Callum Marshall

geboren te Frimley, Het Verenigd Koninkrijk van Groot-Brittannië en
Noord-Ierland
in 1998

Promotores:	Prof.dr. V. Dunjko Prof.dr. T.H.W. Bäck
Co-promotor:	Dr. E.P.L. van Nieuwenburg
Promotiecommissie:	Prof.dr. M. Bonsangue Prof.dr. A. Plaat Dr. H. Basold Prof.dr. S. Jeffery (Universiteit van Amsterdam) Prof.dr. I. Kerenidis (Université Paris Diderot)

Contents

CHAPTER 1

Introduction

In recent decades, quantum algorithms have been discovered that are exponentially faster than our best-known classical algorithms. However, the quantifier *best-known* is needed, as we do not know if quantum computers are fundamentally faster on these tasks or if we just haven't discovered equivalently fast classical algorithms *yet*. This thesis addresses this disconnect, firstly by expanding our theoretical understanding of the consequences if quantum computers and classical computers are equivalently powerful, and then by analysing multiple algorithms which may allow us to make better use of quantum computers deployable today or in the near future.

Chapter 1 provides a set of definitions of the mathematical tools and definitions (such as quantum computing, various complexity classes, and machine learning) we will use in later chapters. For concepts used in only one chapter, we leave definitions to those chapters.

Chapter 2 provides the strongest evidence that quantum computers can implement functions or sampling from distributions that classical computers cannot.

Motivated by understanding the separation between quantum and classical methods in machine learning we develop the concept of bounded advice in Chapter 3, i.e. enhancing some limited Turing machines with advice given by a stronger machine. This concept allows us to understand if quantum computers used to prepare advice (such as during training) can

provide an edge over classical machine learning even when the ultimate model is entirely classical. Bounded advice proves to have utility beyond this, being useful for understanding situations in cryptography and for general inquiry into complexity theory.

The next chapters develop methods to make better use of near-term machines. For the following two chapters we focus on a technique called circuit cutting, which reduces the number of qubits needed for a quantum computation at the cost of an exponentially increasing number of evaluations of quantum circuits.

Chapter 4 proposes a new quantum machine learning algorithm. The algorithm combines many smaller quantum circuits, just as circuit cutting does, but allows for the setting of the number of circuits as a hyperparameter. Lower circuit counts yield an algorithm that could be practically used, higher circuit counts approximate the full circuit cutting scheme. We analyse the learning performance of this model and experimentally test it, this allows us to compare quantum and classical computing more large-scale quantum machines are available.

Chapter 5 continues the above research line by proving that while it may be possible to reduce the number of smaller circuits produced by cutting a larger circuit for certain circuits, this is not the case in general. We prove that if any *cut-local*[1] cutting scheme can remove even a single qubit from a circuit without running an exponential (in number of gates such a cut would entail) number of smaller circuits, then all decision problems solvable by a quantum computer can also be solved by a classical computer.

In the Chapter 6, we develop a quantum machine learning algorithm which can be deployed on a classical computer while still reaping the benefits of a quantum computer to train. This is formally linked to quantum-generated classical advice, as treated in Chapter 2.

[1] this will be defined in Chapter 5

1.1 Quantum Computing

To define quantum computation we first state the postulates of quantum mechanics. Quantum mechanics is commonly defined by 4 postulates. We provide these postulates using bra-ket notation for vectors [1].

1. The state space of any physical system is a Hilbert space. The system is entirely described by a state vector in this Hilbert space.

2. The evolution of a closed quantum system over a fixed amount of time can be described by a unitary transform acting on that system's Hilbert space.

3. The measurement of a system is defined by a set of linear measurement operators $\{M_m\}$. The probability of measuring outcome m for state $|\psi\rangle$ is $p(m) = \langle\psi| M_m^\dagger M |\psi\rangle$. The state after measurement outcome m is $M_m\psi/\sqrt{p(m)}$.

4. The composition of multiple physical systems is given by the tensor product of their associated Hilbert spaces. The composed state of these physical systems is the tensor product of the individual states.

These postulates define the rules of quantum mechanics. This thesis concerns quantum computing, which is an attempt to improve computational efficiency by using the increased capabilities provided by these rules, as compared to those provided by the rules of classical physics.

In quantum computing, we are often only concerned with the quantum bit, as a minimum-sized unit of information which is always a linear combination of 0 and 1. Here we use bra-ket notation and $\alpha, \beta \in \mathbb{C}$ for $|\alpha|^2 + |\beta|^2 = 1$

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Larger systems (such as a computer) can be described as the combination of individual qubits following postulate 4.

Before we define quantum computing, we must define gate sets and circuits. A gate set is a set of unitaries we can apply on our quantum computer, due to the infinite nature of the space of unitaries there is an infinite set of possible gate sets. Any particular quantum device will have some gates that are more natural than others. Fortunately, some gate sets are universal, that is they can approximate any unitary operation on a quantum system to arbitrary precision. For the purpose of theory, it is therefore sufficient to define a single universal gate set and rely on the

ability of other universal gate sets (that may be native to any given device) to approximate our circuit with high accuracy.

The following gates are defined using the vector notation of quantum states ($|0\rangle = (1,0)^T$, $|1\rangle = (0,1)^T$).

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

It is often easiest to describe a quantum circuit pictographically, for single-qubit gates this is normally given by a white box with the name of the gate inside of it. For some 2 qubit gates, we have less direct representations, the CNOT is represented by the following symbol.

(1.1)

The 3 above gates form a universal gate set. Depending on the circumstance it can be easier to write operations with other gates instead, such as with the following "rotation" gates. Of course, these gates could be equally approximated by the above gates, but this would require much more complicated notation for some situations.

$$R_x(\theta) := \quad -\boxed{R_x(\theta)}- \quad = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{1.2}$$

$$R_y(\theta) := \quad -\boxed{R_y(\theta)}- \quad = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{1.3}$$

$$R_z(\theta) := \quad -\boxed{R_z(\theta)}- \quad = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{1.4}$$

1.1.1 Limited modern machines

The previous section described an idealised quantum computation. In the real world, we are not able to perfectly execute quantum circuits; noise is naturally introduced that destroys quantum superposition [1]. Fortunately, if noise is suppressed below a threshold error correction is possible, where many qubits are used to simulate one nearly-perfect or *logical* qubit. Unfortunately, this process of error correction often requires a large number of physical qubits to produce one logical qubit, and a large overhead on the number of physical gates needed to implement one logical gate. Thus it is often considered how we could perform interesting computations while avoiding costly error correction with our modern Noisy Intermediate Scale Quantum computers (NISQ) [2].

There are many interesting approaches to dealing with noise in quantum computing without error correction, but this work instead tackles the other component of NISQ: intermediate scale. In the following subsection, we outline one such method, circuit cutting.

1.1.2 Circuit Cutting

Circuit cutting [3–11], sometimes referred to as circuit partitioning or circuit knitting, is a set of techniques to take one large quantum circuit and to estimate its output by measuring many smaller quantum circuits. Existing schemes either make multiple calls to the device [3–6] or link multiple devices with classical communication [7, 9] to simulate the larger circuit.

Each of the existing circuit-cutting schemes works slightly differently, but they all take a local element of the large circuit and represent it as the sum of terms, which can then be separated into smaller circuits. We will give a generalization of circuit cutting by cutting the unitaries. This generalization involves decomposing a unitary into the sum of several smaller unitaries. Here, decomposing means expressing a given n-qubit unitary as a sum of tensor products of two fewer-qubit unitaries:

$$U = \sum_{i}^{L} \alpha_i U_i' \otimes U_i'', \tag{1.5}$$

As every 2 qubit unitary matrix is equal to the sum of (at most) 4 single qubit tensor product matrices this scheme can break any connecting elements across a large circuit and allow the two chunks to be evaluated independently. This is shown in figure 1.1.

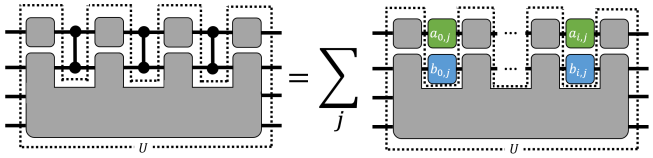


Figure 1.1: Figure depicts circuit cutting applied to a quantum circuit. By separating the three controlled-Z gates into the sum of a polynomial number of single qubit unitary we produce a bottom circuit with $n-1$ qubits and a top circuit consisting of one qubit, which can be run independently, lowering the required number of qubits to evaluate the original circuit.

This circuit-cutting generalisation most closely resembles the original proposal [12]. Other circuit cutting schemes do not necessarily fit this unitary cutting generalisation but cut some other element of the circuit. While these schemes do not fit within the generalisation presented we will see that the theorems we develop for unitary cutting carry over to these cases as they are not conceptually different.

1.2 Machine Learning

Machine learning is a set of statistical methods that take data from some problem and, by adjusting internal weights, are able to generalise to solve that problem at large. There are many fields nested inside machine learning, such as supervised, unsupervised and reinforcement learning. For the sake of this thesis, we will only be interested in supervised learning, i.e. when the problem is to label data in a similar manner to labelled examples.

To define supervised learning we will break it down into a number of separate parts: the task, the model and the training procedure.

Let us begin by formalising the task. We define a supervised learning task on a domain (inputs) $\mathcal{X}$ and co-domain (outputs) $\mathcal{Y}$ with a probability distribution (the probability of each input and output pair) over $\mathcal{X} \times \mathcal{Y}$, P, and loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

To formalise the model, we first define a hypothesis, $h : \mathcal{X} \to \mathcal{Y}$. The hypothesis defines a map from input data points to output labels, this definition does not allow for probabilistic labelling but that can be included in more complicated definitions. Writing out a truth table for h would be cumbersome. Instead, we often define some parameterised model, such as a neural network, that has some set of parameters we can tune, modifying

these parameters gives us access to different hypotheses.

Formalising the training procedure is more complicated as there are many potential training methods in SL (supervised learning). Broadly the training objective is to find parameters that allow the model to perform well on the task. Performing well is often defined by minimising the expected value of the loss function on the output of the model and the correct label. If this distance is small, the model is mostly correct most of the time. So the training procedure's task is to minimise the expected amount of loss[2]. Define the risk for a hypothesis h on a space of continuous inputs/outputs as the expected loss:

$$R(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(h(x), y) dP(x, y). \tag{1.6}$$

The ultimate objective is to minimise this loss.

In practical settings, we normally lack access to the underlying probability distribution, P, so the *true* risk cannot be evaluated. Instead, we have a data set, this data is drawn from P but is only a sample, $S = \{(x_i, y_i) \sim P \,|\, i \in [m]\}$. This dataset makes it possible to approximate the risk, we call this the empirical risk of h with respect to S:

$$\hat{R}_S(h) = \frac{1}{|S|} \sum_{(x_i,y_i)\in S} \ell(h(x_i), y_i). \tag{1.7}$$

Optimising our hypothesis on the training data optimises the empirical risk, which is generally a good proxy for the true risk, but it is possible that these two diverge, this is sometimes called 'generalisation error' and can be characterised by 'over/underfitting'. More about this divergence will be said in Chapter 4 where showing the model defined in that section has a small generalisation loss is of key interest.

1.3 Complexity Theory

Large parts of this thesis are written in terms of complexity classes, loosely defined as sets of problems which can be solved by different levels of computational ability. We assume the reader is familiar with the standard definitions of Turing machines and focus here on establishing the notation and variant models used throughout this work.

[2]It is possible other training objectives could be given.

1.3.1 Computational Models

The *deterministic Turing machine* (DTM) [13] serves as the standard theoretical formalisation of computation. This thesis assumes familiarity with the DTM, but we will also use some standard variants:

A *non-deterministic Turing machine* (NTM) [13] extends the deterministic model by allowing any given state of the Turing machine to lead to a number of possible next states and actions. The NTM is said to accept an input if there exists at least one sequence of non-deterministic choices leading to an accepting state.

A *probabilistic Turing machine* (PTM) [13] builds upon the non-deterministic model by assigning probabilities to each possible transition. While an NTM simply explores all possible computational paths, a PTM randomly selects transitions according to their associated probabilities, providing a framework for randomised algorithms and error bounds in computation. The acceptance criteria for a PTM are given by the particular complexity class it is used in.

A *quantum Turing machine* (QTM) [1] incorporates quantum mechanics into the computational model, allowing configurations to exist in superpositions and employing quantum operations (unitary transformations and measurements) for transitions.

An *oracle Turing machine* [13] augments the standard model with an external 'oracle' that can decide membership in some language L in a single step. These machines are central to defining relativized complexity classes like NP^{A}, where A represents the oracle, and help establish relationships between different complexity classes.

1.3.2 Formal Languages

A *formal language* is a set of strings over a finite alphabet Σ. Formally, a language $L \subseteq \Sigma^*$ is a set of strings that can be recognized by a computational model, such as a Turing machine. A decision problem can be framed as a language by considering the set of all "yes" instances as the strings that belong to the language. For example, the decision problem "Is a number prime?" can be represented as a language where the strings are the binary representations of prime numbers.

1.3.3 Complexity classes

Using the tools of the previous subsections we can now define a 'zoo' of complexity classes.

- P is the class of decision problems, $\{L \subset \{0,1\}^*\}$, that a deterministic Turing machine can determine membership of in time polynomial in the length of x.
- NP is the class of decision problems solvable by a non-deterministic polynomial-time Turing machine that accepts if at least one path accepts.
- CoNP is the complement of NP.
- We define levels of the polynomial hierarch with the following notation $\Sigma_0^P = \Pi_0^P = \Delta_0^P := P$. $\Delta_{i+1}^P := P^{\Sigma_i^P}$. $\Sigma_{i+1}^P := NP^{\Sigma_i^P}$. $\Pi_{i+1}^P := CoNP^{\Sigma_i^P}$[3].
- PH: The polynomial hierarchy is defined as $PH = \bigcup_i \Sigma_i^P$.
- BPP is the class of decision problems solvable by a probabilistic Turing machine with at most a 1/3 probability of error both for $x \in L$ and $x \notin L$.
- ZPP is the zero error alternative to BPP. A language is in ZPP if there exists a probabilistic Turing machine which either refuses to answer (which it does with less than 1/3 probability) or correctly determines if $x \in L$.
- EXP is the class of decision problems solvable by a Turing machine running in $O(2^{p(n)})$ for any polynomial $p(n)$.
- PP is the class of decision problems solvable by a probabilistic Turing machine with a probability of error less than 1/2.
- AM is the class of decision problems for which $x \in L$ can be verified by a probabilistic check: L is in AM if there exists a probabilistic polynomial time verifier, V, such that if $x \in L$ then, with at least 2/3 probability there exists a proof, y, such that V accepts. If $x \notin L$, with at least 2/3 probability, regardless of y, V must reject.
- BQP is the class of decision problems solvable by a quantum Turing machine with at most a 1/3 probability of error.
- QMA is the class of decision problems such that there exists a 'verifier' quantum Turing machine which takes the input and a quantum state 'proof'. For all yes inputs there exists a quantum state that leads the verifier to accept with probability at least 2/3. For all no inputs, all states lead the verifier to reject with probability at least 2/3.

[3]The notation $\_^P$ signals the polynomial hierarchy, not an oracle to P.

- P/poly is the class of decision problems solvable by a family of polynomial-size circuits. Equivalently P/poly is the class of decision problems solvable by a polynomial time deterministic Turing machine with polynomial length *advice*. Advice is an additional input string which may depend on the size of the input, but not on the input itself. Chapter 3 will provide more information on advice and it's associated concepts.

In complexity theory, a functional problem is defined by a relation, R. An algorithm solves this problem if, for all inputs, x, the algorithm outputs y such that $(x, y) \in R$ or halts if no such y exists. As with decision classes, there are a number of functional complexity classes we will need to define.

- #P is the class of functional problems that consist of computing the number of accepting paths of a non-deterministic Turing machine.
- FBPP is the class of functional problems that can be solved by a polynomial time probabilistic Turing machine which achieves any failure probability $\epsilon > 0$ given input $x, 0^{\lceil 1/\epsilon \rceil}$.
- FBQP is the class of functional problems that can be solved by a polynomial-time quantum Turing machine which achieves any failure probability $\epsilon > 0$ given input $x, 0^{\lceil 1/\epsilon \rceil}$[4].

While it is well known that functional and sampling problems are closely related [14], we will still make reference to the following sampling problems. A sampling problem is defined by a function that maps each input, x, to a probability distribution D_x. An algorithm solves the problem if, on input x, it outputs y sampled from close to D_x (where close is defined for each individual class).

- SampP is the class of sampling problems, $S = (\mathcal{D})_{x \in \{0,1\}^*}$ for which there exists a polynomial-time probabilistic Turing machine which given input $x, 0^{\lceil 1/\epsilon \rceil}$ outputs a distribution ϵ close to the problem's distribution in total-variation distance.
- SampBQP is the class of sampling problems for which there exists a polynomial-time quantum Turing machine which given input $x, 0^{\lceil 1/\epsilon \rceil}$ outputs a distribution ϵ close to the problem's distribution in total-variation distance.

[4] the notation 0^n refers to a string of n 0s

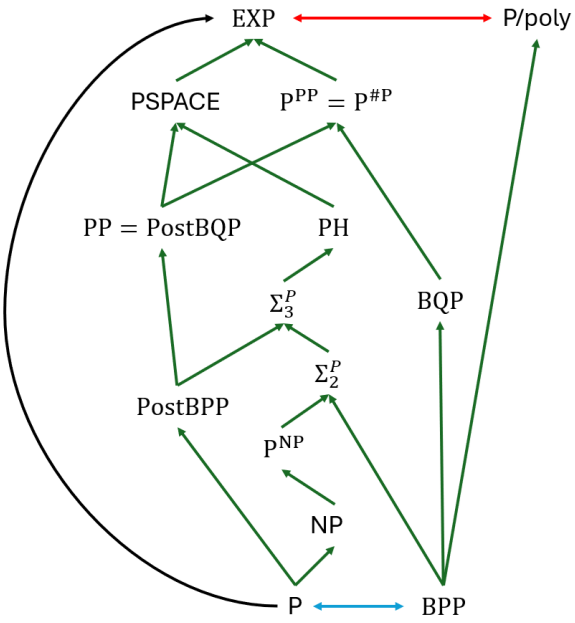


Figure 1.2: A diagrammatic representation of various complexity classes which contain each other. $\rightarrow$ represents a known inclusion that we believe to be strict. $\leftrightarrow$ represents classes we believe are unequal, while $\leftrightarrow$ represents two classes we believe to be equal. $\rightarrow$ is a known strict inclusion.

One particularly relevant problem in SampBQP is BosonSampling. This problem can be defined with reference to a *Bosonic computer* [15], but it is perhaps clearer to use the equivalent mathematical definition. For a matrix $A \in \mathbb{C}^{m \times n}$ and string of numbers $x \in \mathbb{N}^m$ such that $\sum x_i = n$, define the submatrix A_x as the $n \times n$ matrix made by repeating the ith row of A x_i times. The target distribution of BosonSampling is dependent on the permanents of these submatrices. On input A, a column orthonormal complex matrix, the string $x \in \mathbb{N}^m$ should be sampled probability $Per(A_x)/x_0!x_1! \dots x_m!$.

This finishes the definitions necessary for this thesis. It is also noteworthy to explore connections between these classes, a quick representation is given in Figure 1.2. Of particular note to this thesis are the questions of $\mathsf{BQP} \stackrel{?}{=} \mathsf{BPP}$ or $\mathsf{FBQP} \stackrel{?}{=} \mathsf{FBPP}$, which ask whether quantum computers can solve decision problems or functional/search problems that classical

probabilistic computers cannot.

1.3.4 Proof Relativisation

In chapter 2 we will note that our particular style of proof contains insight beyond it's literal results. Namely, our proofs overcome the so called 'relativisation barrier' [16], and as such, they may serve as a starting point for further advancements in complexity theory. While a full description of the numerous barriers in complexity theory is outside the scope of this thesis, we can provide a brief introduction to the relativisation barrier.

The relativisation barrier is a concept used by complexity theorists to determine if a certain style of proof can be used to establish a result, based on whether that result holds (or fails to hold) relative to various oracles. To illustrate, suppose we are trying to prove two complexity classes are not equal, e.g. P and NP. If we know there exists an oracle A such that $\mathsf{P}^{\mathsf{A}} = \mathsf{NP}^{\mathsf{A}}$ and a second oracle B such that $\mathsf{P}^{\mathsf{B}} \neq \mathsf{NP}^{\mathsf{B}}$, then any proof of $\mathsf{P} \neq \mathsf{NP}$ must somehow not function when the classes are made relative to A [16].

For a number of results, we know the result does not hold relative to all oracles, thus we know the proof of the result must be 'non relativising'. This makes proof techniques which 'cross the relativisation barrier' very exciting for proving non relativising results.

1.4 Research questions

It is useful to frame this thesis with a number of research questions.

Research question 1
What is the strongest theoretical basis for the claim "In polynomial time quantum computers can perform computations that classical computers cannot"?

The first chapter to address this question is Chapter 2, which approaches from a complexity theoretic angle. Particularly, we see what unexpected complexity theory collapses might occur if FBQP is equal to FBPP. This chapter finds if $\mathsf{FBQP} = \mathsf{FBPP}$ then the polynomial hierarchy collapses to the second order. The strongest known collapse for any implementable quantum class. Chapter 3 asks subquestions to this research question, first asking if quantum computers can outperform classical on the task of generating advice (in the sense of $\mathsf{P/poly}$), then asking if quantum computers can outperform classical computers when receiving advice. Chapter

3 demonstrates the importance of these questions to the questions of $\mathsf{BQP} \stackrel{?}{=} \mathsf{P/poly}$ and $\mathsf{BQP} \stackrel{?}{=} \mathsf{BPP/samp}$. The answers to these subquestions naturally lead to the next research question.

Research question 2
If polynomial-time quantum computers can give better advice than polynomial-time classical computers, can we find such an advice-generating algorithm?

This research question is the topic of Chapter 6, which describes a machine learning algorithm which uses the quantum computer to find a good set of weights (the advice in this case), but which can then be deployed purely on a classical computer for any input. Indeed, we show that this algorithm captures all other surrogate models and is in some sense universal for this type of problem.

The machine learning model developed in Chapter 6 is naturally more suited to modern quantum computers because it does not require the use of an expensive quantum computer for each evaluation. This is a useful property to have and leads us to ask if we can further reduce the burden of quantum machine learning by reducing the requirements on the size of our quantum computer.

Research question 3
Do there exist methods to reduce the number of qubits required to run a given machine learning algorithm?

Chapter 4 first provides evidence that for some circuits this may be possible. It develops a quantum machine learning algorithm which can naturally simulate larger quantum circuits than those which are being used to deploy the quantum algorithm, it does this while also presenting a path to reduce the runtime of this simulation when such an option is available. Numerical evidence is then used to show that there exist examples where this algorithm is indeed capable of reducing qubit counts, by learning the output of a random quantum circuit using circuits of half the size. However, Chapter 5 shows that not all circuits can be reduced by this (or any other) method. It shows that there exist circuits such that the ability to remove even one qubit efficiently (lowering the number of qubits needed by even one) would imply $\mathsf{BQP} = \mathsf{BPP}$.

CHAPTER 2

Improved separation between quantum and classical computers for sampling and functional tasks

Quantum computers are thought to be poised to outperform classical computers on a number of significant tasks, such as integer factorisation or simulation of quantum systems. Despite this widely held belief, we have no formal proof that factoring is not solvable in polynomial time on a classical computer, or even that $\mathsf{BQP} \neq \mathsf{BPP}$.

This is perhaps not surprising given the challenge posed by proving unconditional results in complexity theory; famously $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ remains unresolved after decades of study. A more realistic approach is to base the hardness of quantum computing on widely accepted complexity-theoretic conjectures.

This approach has so far been quite successful for sampling problems, with a number of papers [17, 18] showing that various quantum sampling experiments would be hard for a classical algorithm unless the polynomial hierarchy collapses to its third level, or else a widely believed conjecture fails. These results are especially interesting as they naturally imply that quantum computers could implement functions classical computers could not ($\mathsf{FBQP} \neq \mathsf{FBPP}$) due to a surprising equivalence between sampling and functional classes [14].

Similarly, there has been a body of work showing that various changes to quantum computing make quantum computing vastly more powerful

than similarly modified classical computing. A well-known example is PostBQP vs PostBPP, i.e. quantum/classical computing with postselection. Surprisingly, the former is in some sense equivalent to problems involving exact counting (PP) [19], while the latter is only equivalent to approximate counting [20]. This disconnect implies that the classes are likely not equal, indeed if they are this would also imply a collapse of the polynomial hierarchy to the third level [19].

Both of these research lines provide strong evidence for that quantum computing is more powerful than classical computing, but they rely on a number of conjectures and complexity theory conditions. We are therefore motivated to attempt to minimise or remove the need for these conjectures/conditions until we are left with compelling evidence that quantum computing is fundamentally stronger than classical computing.

Hope that weaker conditions might be possible was provided by Fujii et al. [21], who showed that if classical computers can *exactly sample* from any one of a number of quantum advantage experiments then the polynomial hierarchy would collapse to its second level, an improvement over the existing collapse to third level. In fact they push the collapse all the way to AM, in the second level of the hierarchy. While this result provides insight as to what may be possible, it stops short of our goal as the proof strategy did not extend to additive approximate sampling, which is arguably the realistic case [17] and is relevant for questions such as $\mathsf{FBQP} \stackrel{?}{=} \mathsf{FBPP}$.

In this chapter, we provide the following new complexity theoretic result, which we then use to improve the abovementioned results to a 2nd level polynomial hierarchy collapse. In the following theorem, we use $\|$ to denote only one round of parallel oracle calls (i.e. many oracle calls can be asked, but must all be asked in a single round, without adapting to the previous oracle calls).

Theorem (Main)
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

By showing that the existing collapse results for classical boson sampling, commuting circuit sampling and 1 clean qubit sampling can all be modified to show $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ (we are required to add the parallel requirement to $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ for our collapse), we improve the implied collapse of the polynomial hierarchy from 3rd order to 2nd. Consequently, this strengthens the evidence that $\mathsf{FBQP} \neq \mathsf{FBPP}$. This theorem can also be made to strengthen the hierarchy collapse implied by assuming $\mathsf{PostBQP} = \mathsf{PostBPP}$ from the third to the second level.

From a purely complexity-theoretic perspective, there is an interesting alternative form of this result in terms of exact and approximate counting[1].

$$\text{If } \mathsf{P}_{\|}^{\mathsf{ExactCount}} = \mathsf{P}_{\|}^{\mathsf{ApproxCount}} \text{ then } \mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$$

This naturally gives our main result the following interpretation: *If the problems solved with the aid of exact counting can also be solved with approximate counting, then both are contained in* $\mathsf{ZPP}^{\mathsf{NP}}$. The generality of this interpretation hints that there may be applications of this theorem to counting beyond the quantum theory considered herein. It is also interesting to note that the proof contained herein does not relativise as we rely on the checkability of $\#\mathsf{P}$ for a step of our proof.

The chapter is structured in 3 sections. Section 2.1 focuses on the proof of our central theorem, section 2.2 proves that the polynomial hierarchy collapses to second order if $\mathsf{PostBQP} = \mathsf{PostBPP}$. In the third section, we prove our various sampling results. Finally, we close by discussing interesting future research and open questions.

2.1 Main theorem

In this section we prove our main theorem:

Theorem 2.1.1
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$.

We will assume readers are broadly familiar with classes such as $\#\mathsf{P}$, where classes are undefined we use the definitions given in the complexity zoo [22]. We use the notation $\mathsf{A}_{\|}^{\mathsf{B}}$ for A with one set of parallel oracle calls to B.

It is useful to first give an informal description of the proof of theorem 2.1.1 before we proceed to the formal proof to give the reader a better idea of the purpose of each lemma. Broadly the proof relies on the notion of random-reducibility, where a given instance of a problem can be solved by a polynomial time algorithm with randomly selected instances of some other problem. If a problem can be randomly reduced to itself we say it is random self-reducible. The first key realisation for Theorem 2.1.1 is that a random self-reducible language that is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ is itself random reducible to NP, because randomly selecting x for a language in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ gives rise

[1] Exact counting can be defined as exactly counting the number of accepting inputs of some poly-time machine, approximate counting is getting within a multiplicative factor of 2 of exact counting (equivilently to within a $1 + \frac{1}{poly(n)}$ factor).

to random set of non-adaptive calls oracle calls to NP, which meets the definition of random reducibility to NP. Next, as #P is random self-reducible [23], the antecedent of Theorem 2.1.1 implies that it is random reducible to NP. The second key realisation is to notice that the proof by Feigenbaum and Fortnow that NP cannot be random self-reducible unless it is also in coNP/poly extends beyond self reducibility; any language that is random reducible to NP is in coNP/poly. By repeating the argument for coNP we show $\#\mathsf{P} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Arvind and Köbler [24] showed any checkable language in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ is low for Σ_2^{P} which gives us a polynomial hierarchy collapse to Σ_2^{P}. The final collapse to $\mathsf{ZPP}^{\mathsf{NP}}$ comes from using the $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ algorithm to produce proofs verifiable in P^{NP}, thereby achieving zero error.

We begin by defining random reducibility. A problem is randomly self-reducible if any given instance of the problem can be solved probabilistically by a polynomial time algorithm with access to solved random instances of the same problem. For our purposes we will work with a similar concept: random-reducibility, which is the same except the random instances may be from a different problem. It should be stated that this is different from a 'randomized reduction', which is a randomised Karp reduction. Random reductions are sometimes called 1-locally random reductions to avoid this confusion.

Definition 2.1.1 (Random Reducible)
*A function f is **nonadaptively** $k(n)$ **random-reducible** to a function g if there are polynomial time computable functions σ and ϕ with the following two properties.*

(1) For all n and all $x \in \{0,1\}^n$ $f(x)$ can be solved by ϕ with instance of $g(y)$ for y selected by σ with high probability:

$$\Pr_r \left(f(x) = \phi(x, r, g(\sigma(1, x, r)), \ldots, g(\sigma(k, x, r)))\right) \geq 2/3$$

(2) For all n, all pairs $\{x_1, x_2\} \subset \{0,1\}^n$ and all i, if r is chosen uniformly at random then $\sigma(i, x_1, r)$ and $\sigma(i, x_2, r)$ are identically distrubuted.

If both conditions hold we say (σ, ϕ) is a random-reduction from f to g.

As we are dealing with no other forms of random reducibility we will just say 'f is rr to g' when f is non-adaptively $k(n)$ random-reducible to g for some polynomial k. If a function is rr to itself then we say the function is random self-reducible (rsr). A set is rr to another if its characteristic

function is rr. A set of languages, A is rr to B if every $L \in \mathsf{A}$ is rr to some $L' \in \mathsf{B}$.

As with probabilistic classes like BPP, we can boost random reducibility an arbitrarily low (2^{-n}) failure probability.

Lemma 2.1.1 ([23])
If function f is non-adaptively $k(n)$ random-reducible to g then for any polynomial, $t(n)$, f is non-adaptively $24t(n)k(n)$ random-reducible to g where condition (1) holds for at least $1-2^{-t(n)}$ of the r's in $\{0,1\}^{24t(n)k(n)}$ (probability $1-2^{-t(n)}$ of success).

Remark
In general, this boosting may not work for a definition of random reducibility dealing with relations instead of functions as they may have multiple correct outputs. However, counting problems like Perm, which has only one correct output for a given input, can be boosted.

Lemma 2.1.2 ([23])
Any $\#\mathsf{P}$-complete language is rsr.

This concludes our definitions, we can now state an intermediate theorem to our final result.

Theorem 2.1.2
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP.

Before we prove this, we will provide a number of smaller results. In the following lemma the notation $\mathsf{A}^{(\mathsf{B}[1])}$ means A with one oracle call to B.

Lemma 2.1.3
Any language in $\mathsf{P}^{\#\mathsf{P}[1]}$ is random reducible to $\#\mathsf{P}$.

Proof. As $\#\mathsf{P}$-complete languages are random self-reducible and only one oracle call is needed we can use a random reduction of $\#\mathsf{P}$ to give the answer to the oracle call and use the polynomial time ϕ algorithm to perform the rest of the P algorithm. □

The next lemma captures the intuition that $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ algorithms are 'almost' rr to NP, only missing the random element selection part of the definition (property (2)).

Lemma 2.1.4
For all L in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ there exists polynomial time computable functions σ_B, ϕ_B such that

$$\Pr_r\left(L(x) = \phi_B(x, r, \mathsf{SAT}(\sigma_B(1, x, r)), \ldots, \mathsf{SAT}(\sigma_B(m, x, r)))\right) \geq 1 - 2^{-n}.$$

Proof. If $L \in \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then there exists a polynomial time algorithm, A, to decide $x \in L$ which calls only one set of up to m non-adaptive NP queries. We assume the NP calls SAT for simplicity. Let the algorithm for $\sigma_B(i, x, r)$ run A until the step involving oracle queries and then output just the i'th query (assuming some arbitrary query ordering). The algorithm for ϕ_B is now just the algorithm A using the oracle calls produced asked by $\sigma_B(i, x, r)$ in place of directly making its own calls. Since both ϕ_B and σ_{B} have access to the same random string r they will both select the same oracle calls. □

Lemma 2.1.4 is useful as it proves that $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ fulfils property (1) of the definition of random reducibility to NP, but does not prove the random distribution of $\sigma_B(1, x, r)$. The next theorem shows that if a langauge already randomly reduces to $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ then this random reduction condition (condition (2)) is also fufilled, and thus the language is random reducible to NP

Lemma 2.1.5
All languages random reducible to a language in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ *are random reducible to* NP.

Proof. Let L be a language which is random reducible to $L_B \in \mathsf{BPP}_{\|}^{\mathsf{NP}}$. We will demonstrate that L is rr to NP by writing out the definition of random reducibility to L_B then using lemma 2.1.4 to substitute the calls to L_B with a formula using calls to SAT. We can then rearange this into a format which directly makes calls to SAT and we show that these calls inherit the randomness property from the random reduction of L to L_B.

Let us assume the random reduction of L to L_B on input x uses k calls[2], from the definition of random reducibility there exists ϕ and σ such that

$$\Pr_r \left(L(x) = \phi(x, r, L_{\mathsf{B}}(\sigma(1, x, r)), \ldots, L_B(\sigma(k, x, r)))\right) \geq 1 - 2^{-n}.$$

Define $y_i := \sigma(i, x, r)$.

$$\Pr_r \left(L(x) = \phi(x, r, L_{\mathsf{B}}(y_1), \ldots, L_{\mathsf{B}}(y_k))\right) \geq 1 - 2^{-n}$$

Using lemma 2.1.4 we can substitute L_B with ϕ_B, σ_B and a random string r_i for the i'th call to L_B. In the following we assume each lemma-2.1.4-reduction uses m SAT calls.

[2] an input of length n can only use some polynomial number of oracle calls, upper bounded by k

$$\Pr_{r,r_1,\ldots r_k}\big(L(x) = \phi(x,r,\phi_B(y_1,r_1,\mathsf{SAT}(\sigma_B(1,y_1,r_1)),\ldots,\mathsf{SAT}(\sigma_B(m,y_1,r_1))),$$
$$\ldots,\phi_B(y_k,r_k,\mathsf{SAT}(\sigma_B(1,y_k,r_k)),\ldots,\mathsf{SAT}(\sigma_B(m,y_k,r_k))))\big)$$
$$\geq 1-(k+1)2^{-n}.$$

The failure probability $1-(k+1)2^{-n}$ comes the failure probability of the k reductions combined with the failure probability from the random reduction.

We can now begin to combine elements of the reduction from L to L_B with reduction from L_B to SAT. We start with ϕ which we combine with ϕ_B to define ϕ', informally we can think of this as doing the two polynomial-time algorithms as a larger polynomial time algorithm.

$$\phi'(x,r,r_1,\ldots,r_k,\mathsf{SAT}(\sigma_\mathsf{B}(1,\mathsf{y}_1,\mathsf{r}_1)),\ldots\mathsf{SAT}(\sigma_\mathsf{B}(\mathsf{m},\mathsf{y}_\mathsf{k},\mathsf{r}_\mathsf{k}))) :=$$
$$\phi(x,r,\phi_B(y_1,r_1,\mathsf{SAT}(\sigma_B(1,y_1,r_1)),\ldots,\mathsf{SAT}(\sigma_B(m,y_1,r_1))),\ldots,$$
$$\phi_B(y_k,r_k,\mathsf{SAT}(\sigma_B(1,y_k,r_k)),\ldots,\mathsf{SAT}(\sigma_B(m,y_k,r_k))$$

We use '$x\#y$' to denote y appended to x. Define $\mathbf{r} := r\#r_1\#\ldots\#r_k$.

Again we define a new function, σ', as the combination of two existing function, σ and σ_B:

$$\sigma'(i,j,x,\mathbf{r}) := \sigma_B(i,\sigma(j,x,r),r_j).$$

Which gives the following simplified equation:

$$\Pr_{\mathbf{r}}\big(L(x) = \phi'(x,\mathbf{r},\mathsf{SAT}(\sigma'(1,1,x,\mathbf{r})),\ldots\mathsf{SAT}(\sigma'(m,k,x,\mathbf{r}))\big) \tag{2.1}$$
$$\geq 1-(k+1)2^{-n}. \tag{2.2}$$

We will now directly check the definition of σ', ϕ' and the equation 2.1 against the definition of random-reducbile, recall the two conditions definition 2.1.1:

(1) For all n and all $x \in \{0,1\}^n$ $f(x)$ can probably be solved by ϕ with instance of $g(y)$ for y selected by σ:

$$\Pr_r\big(f(x) = \phi(x,r,g(\sigma(1,x,r)),\ldots,g(\sigma(k,x,r)))\big) \geq 2/3$$

(2) For all n, all pairs $\{x_1,x_2\} \subset \{0,1\}^n$ and all i, if r is chosen uniformly at random then $\sigma(i,x_1,r)$ and $\sigma(i,x_2,r)$ are identically distrubuted.

We can see that these two conditions are met:

(1) For sufficiently large n, $1-(k+1)2^{-n} > 2/3$ therefore equation 2.1 is of the form:

$$\Pr_r\left(f(x) = \phi(x, r, \mathsf{SAT}(\sigma(1, x, r)), \ldots, \mathsf{SAT}(\sigma(k, x, r)))\right) \geq 2/3.$$

(2) In short: a composition of i.d. maps is an i.d. map. For all n, all pairs $\{x_1, x_2\} \subset \{0,1\}^n$, all i and j, if r is chosen uniformly at random then $\sigma(j, x_1, r)$ and $\sigma(j, x_2, r)$ are identically distributed (as per their definition), therefore $\sigma'(i\#j, x_{1/2}, \mathbf{r})$ is identically distributed too. This is because the only dependence on x is through the identically distributed σ: $\sigma'(i\#j, x, \mathbf{r}) = \sigma_{\mathsf{B}}(i, \sigma(j, x, r), r_j)$. This holds for all r_i, r_j.

Thus fufilling the conditions for L to be random reducible to NP □

Finally, we can proceed with the proof of theorem 2.1.2

Proof of theorem 2.1.2. Toda [25] showed that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}[1]}$. Therefore, by the assumption of the theorem if $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}} = \mathsf{PH}$ then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\#\mathsf{P}[1]}$.

By lemma 2.1.3 we know $\mathsf{P}^{\#\mathsf{P}}$ is rr to $\#\mathsf{P}$. Under the assumption of this theorem, $\#\mathsf{P} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$, $\mathsf{P}^{\#\mathsf{P}}$ is rr to $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. By lemma 2.1.5 this means $\mathsf{P}^{\#\mathsf{P}}$ is rr to NP. □

The next result is heavily based on the work of Feigenbaum and Fortnow [23]. They show that if NP is random self reducible then coNP is in $\mathsf{NP/poly}$, while the result we are trying to show is slightly different than this, the method is very similar.

Theorem 2.1.3
Any language that is random-reducible to a language in NP *is also in* $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$.

Proof. Let $\mathsf{AM}^{\mathsf{poly}}$ be AM with polynomial advice given to Arthur[3]. It turns out that $\mathsf{AM}^{\mathsf{poly}} = \mathsf{NP/poly}$ and $\mathsf{coAM}^{\mathsf{poly}} = \mathsf{coNP/poly}$ [23]. We will prove L is in $\mathsf{AM}^{\mathsf{poly}} \cap \mathsf{coAM}^{\mathsf{poly}}$, beginning with $L \in \mathsf{AM}^{\mathsf{poly}}$.

Suppose $L(x)$ is non-adaptively $k(n)$ random-reducible to $\mathsf{SAT}(x)$ with error probability 2^{-n}. We will adopt the notation $y_i = \sigma(i, x, r)$. For

[3]This is not the same as $\mathsf{AM/poly}$, as the latter must be an AM language for all advice, whereas the former only needs to be defined for the correct advice.

instance size n we will give Arthur the advice $(p_1, \ldots, p_k)$ where p_i is the probability that $y_i = 1$,

$$p_i = \Pr_r(\mathsf{SAT}(\sigma(i, x, r)) = 1).$$

As $\sigma(i, x, r)$ is distributed identically (given uniform random r) for all x this advice does not depend on the input x.

The proof system will consist of Arthur selecting $m = 9k^3$ random strings, $\{r_j\}_{j\in[m]}$, and passing this to Merlin, these random strings defines which SAT queries, $y_{0,j}, \ldots, y_{k,j}$, will be made. Merlin will hand back m 'transcripts'. These transcripts consist of a list $w_{i,j}$ which is either a witness for $y_{i,j}$ or is the string 'NIL' (which is interpreted as the claim that $y_{i,j} \notin \mathsf{SAT}$). For each of the m random strings we get the transcript:

$$Transcript(x, r_j) = (w_{0,j}, w_{1,j}, \ldots, w_{k,j})$$

Arthur performs three checks:

(1) For all i, j either the witness $w_{i,j}$ is 'NIL' or he checks the witness is valid for $y_{i,j} \in \mathsf{SAT}$

(2) Let $b_{i,j}$ be 0 if $w_{i,j}$ is 'NIL' and one otherwise. For all $j \in [m]$ Arthur checks that $\phi(x, r_j, b_{0,j}, \ldots, b_{k,j}) = 1$, i.e. If Merlin has told the truth then ϕ will accept.

(3) For each $i \in [k]$ Arthur checks that more than $p_i m - 2\sqrt{km}$ of the $y_{i,j}$ have been proved to be in SAT, if this condition does not hold he rejects.

If these checks all pass, then Arthur accepts. The trick in this proof is this final condition. Over the random strings, each y_i has some probability of being in SAT with a given variance, by picking m to be large we force this variance to be small. With high probability, a correct answer lies in this range. Since Merlin cannot lie about yes instances (since he must prove them), he can only lie about no instances. However, if he lied too much it would be clear that not enough of y_i are in SAT, thus we could probabilistically guess he was cheating. To exploit this, m is picked precisely so Merlin cannot cheat on all j without exceeding his 'lying budget'. We will now formally show soundness and completeness.

Completness. If $x \in L$ and Merlin is honest, providing witnesses for all $y_{i,j}$ in SAT, condition (1) will always hold. Condition (2) holds with probability at least $1 - 2^{-n}$ (by the definition of rr). Therefore we just need to show condition (3) holds with high probability.

Let $Z_{i,j} := (y_{i,j} = 1)$ and $Z_i = \sum_{j=1}^{m} Z_{i,j}$. We defined our advice so $p_i = \mathbb{E}[Z_i]/m$, we can derive that $\mathrm{Var}(X) = p_i(1-p_i)m < m$. We can use these properties and Chebyshev's inequality to bound our probability of failing check (2).

$$\begin{aligned}
&\Pr\Big(Z_i \leq p_i m - 2\sqrt{km}\Big) \\
&\leq \Pr\Big(\|Z_i - p_i m\| \geq 2\sqrt{km}\Big) \\
&= \Pr\Big(\big\|Z_i - \bar{Z}_i\big\| \geq 2\sqrt{km}\Big) \\
&\leq \mathrm{Var}(X)/4km \leq 1/(4k) \leq 1/4
\end{aligned}$$

The probability of failing any check given $x \in L$ is less than $\frac{1}{4} + 2^{-n}$ which is less than $1/3$ for large enough n. This proves completeness

Soundness. Suppose $x \notin L$, if Arthur accepts then all 3 checks must have passed for all $j \in [m]$. If some $y_{i,j}$ is in SAT but Merlin has provided $w_{i,j} = `NIL'$ then we say Merlin has lied about $y_{i,j}$, if check (1) passes Merlin has not lied about any 'yes' instances, so we disregard this case. The probability of the random reduction failing is 2^{-n} without any lies, so for all m reductions to fail Merlin must lie m times with probability $(1-2^{-n})^m > 1 - m2^{-n}$.

If Merlin lies m times there must be an i that he claims at least m/k of the $y_{i,j}$ are not in SAT when they are. To satisfy condition (3) at least $p_i m - 2\sqrt{km} + \mathbf{m/k}$ of the $y_{i,j}$ must be in SAT to leave 'room' for Merlin to lie m/k times without violating (3). The probability of this is given by a Chernoff bound on the random variable Z_i, as defined above.

$$\begin{aligned}
&\Pr\left(Z_i \geq p_i m - 2\sqrt{km} + m/k\right) \\
&= \Pr\left(Z_i - p_i m \geq m/k - 2\sqrt{km}\right) \\
&= \Pr\left(Z_i - p_i m \geq 9k^3/k - 2\sqrt{k9k^3}\right) \\
&= \Pr\left(Z_i - p_i m \geq 3k^2\right) \\
&\leq e^{-2\times 9k^4/9k^3} = e^{-2\times k} \\
&\leq 1/(4k)
\end{aligned}$$

Combining this with the normal probability that the random reduction fails even with correct oracle answers (which had a 2^{-n} probability) gives an acceptance probability given $x \notin L$ of less than $m2^{-n} + 1/(4k) < 1/3$

for large enough n. This proves soundness.

The previous analysis can just as easily be repeated for $\mathsf{coAM}^{\mathsf{poly}}$ with Merlin proving no instances, giving $L \in \mathsf{AM}^{\mathsf{poly}} \cap \mathsf{coAM}^{\mathsf{poly}}$. By Feigenbaum and Fortnow we know this equals $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. □

To prove the next Theorem we must recall a crucial result by Arvind and Köbler: checkability [22, 24]. The notion of checkability is somewhat involved but intuitively has to do with whether efficent programs that decide the set can themselves be efficently checked. Since we just need the fact that PP and #P are checkable to deploy the following theorem we will not formally define checkability, instead we refer the reader to [24, 26].

Theorem ([24], Theorem 22)
Checkable sets in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ *are low for* Σ_2^{P}.

The last result may seem unnecessary as it is well known that all sets which are in $\mathsf{NP} \subset (\mathsf{NP} \cap \mathsf{coNP})/\mathsf{poly}$ are also in $\mathsf{ZPP}^{\mathsf{NP}}$ [27], however the proof of this does not seem to carry over to $\mathsf{NP} \subset (\mathsf{NP/poly}) \cap (\mathsf{coNP/poly})$, and only with checkability can we get our set in $\mathsf{ZPP}^{\mathsf{NP}}$

Lemma 2.1.6
If $\mathsf{P}^{\#\mathsf{P}} \subset (\mathsf{coNP/poly} \cap \mathsf{NP/poly})$ *then* $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Proof. By Toda $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$ [25]. If P^{PP} is in $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$ then clearly so is PP, combining this with the existence of PP-complete checkable sets [24] we know that PP is low for Σ_2^{P}.

This lowness implies we can put $\mathsf{P}^{\#\mathsf{P}}$ in Σ_2^{P} by the following series of inclusions

$$\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}} \subseteq (\Sigma_2^{\mathsf{P}})^{\mathsf{PP}} = \Sigma_2^{\mathsf{P}}.$$

As $\Sigma_2^{\mathsf{P}} \subseteq \mathsf{P}^{\#\mathsf{P}}$ by Toda [25] we can fix the previous inclusion into an equality: $\Sigma_2^{\mathsf{P}} = \mathsf{P}^{\#\mathsf{P}}$.

As $\Pi_2^{\mathsf{P}} \subseteq \mathsf{P}^{\#\mathsf{P}}$ (Toda [25]) we get $\Pi_2^{\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}}$. Which gives us the final equality:

$$\mathsf{P}^{\#\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}} \subseteq \mathsf{P}^{\Pi_2^{\mathsf{P}}} = \mathsf{P}^{\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$$

□

The previous lemma has achieved a collapse to the second level but we can collapse to $\mathsf{ZPP}^{\mathsf{NP}}$ by using the proveability of languages in $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ to make the $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ algorithm zero-error.

Lemma 2.1.7
All languages in $\mathsf{BPP}^{\mathsf{NP}} \cap \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ *are in* $\mathsf{ZPP}^{\mathsf{NP}}$

Proof. Fix some $L \in \mathsf{BPP}^{\mathsf{NP}} \cap \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$. We can check if $x \in L$ using the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm. Use the $\mathsf{BPP}^{\mathsf{NP}}$ program to determine a proof of this fact for either the Σ_2^{P} verifier or the Π_2^{P} verifier. We can test either proof in P^{NP} using the verifiers from either Σ_2^{P} or Π_2^{P}. With probability $1 - 2^{-n}$ the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm succeeds in producing a valid proof in polynomial time, any valid proof proves or disproves $x \in L$. Therefore in polynomial time the algorithm has successfully determined if $x \in L$ with probability $1 - 2^{-n}$ and never incorrectly answers, placing $L \in \mathsf{ZPP}^{\mathsf{NP}}$. □

This completes the proof of our main theorem which is given below.

Theorem
If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

Summary of proof. If $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP (Theorem 2.1.2).

If $\mathsf{P}^{\#\mathsf{P}}$ is random reducible to NP then $\mathsf{P}^{\#\mathsf{P}}$ is in $\mathsf{coNP/poly} \cap \mathsf{NP/poly}$ (Theorem 2.1.3).

If $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{coNP/poly} \cap \mathsf{NP/poly}$ then $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ (Lemma 2.1.6).

If $\mathsf{P}^{\#\mathsf{P}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ and $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ (Lemma 2.1.7). □

Remark
Interestingly we can perform all of the oracle calls required to produce the proof of $x \in L$ in one parallel step, it then only takes one extra Oracle call to check this proof. Thus it is possible to answer correctly with probability $1 - 2^{-n}$ or with 'I don't know' after only 2 rounds of parallel NP *oracle calls.*

Remark
Feigenbaum and Fortnow [23] showed their proof applies beyond non-adaptive random self-reducibility, up to logarithmically many adaptive steps are allowed. For the same reasons, our proof could be extended to logarithmically many rounds of polynomially many parallel oracle calls.

The following theorem formalises the 'natural interpretation' of our result given in the intro.

Theorem 2.1.4 (Natural interpretation)
If $\mathsf{P}_{\|}^{\mathsf{ExactCount}} = \mathsf{P}_{\|}^{\mathsf{ApproxCount}}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$

This result becomes easy to show after Theorem 2.2.1, so we will prove it at the end of the next section.

2.2 PostBQP and PostBPP

An immediate consequence of the last section's main theorem is to the question of $\mathsf{PostBQP} \stackrel{?}{=} \mathsf{PostBPP}$, i.e. is a poly-time quantum computer with postselection equal to a classical computer with postselection? The best existing answer was given when Aaronson showed $\mathsf{PostBQP} = \mathsf{PP}$ [14], thus equality would imply $\mathsf{PH} = \mathsf{BPP}^{\mathsf{NP}}$ (a collapse to the third level). However, as $\mathsf{PostBPP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ [20, 28] our theorem can be used to improve this collapse to the second level. Before we provide proof of this we formally define PostBQP and the operator $\mathsf{BP}\cdot$ which makes a complexity class probabilistic.

Definition 2.2.1
PostBQP *is the set of languages for which there exists a uniform family of polynomial width and polynomial depth quantum circuits* $\{C_n\}_{n\geq 1}$ *such that for any input* x*,*

(i) There is a non-zero probability of measuring the first qubit of $C_n |0\ldots0\rangle |x\rangle$ *to be* $|1\rangle$*.*

(ii) If $x \in L$*, conditioned on the first qubit being* $|1\rangle$*, the second qubit is* $|1\rangle$ *with probability at least 2/3.*

(iii) If $x \notin L$*, conditioned on the first qubit being* $|1\rangle$*, the second qubit is* $|1\rangle$ *with probability at most 1/3.*

PostBPP can be defined similarly with classical circuits and additional input of random bits.

Definition 2.2.2
Let K *be a complexity class. A language* L*, is in* $\mathsf{BP}\cdot\mathsf{K}$ *if there exists a language* $\mathsf{A} \in \mathsf{K}$ *and a polynomial* p *such that for all strings* x*.*

$$Pr(r \in \{0,1\}^{p(|x|)} : x \in L \iff x\#r \in A) \geq 2/3$$

This covers the necessary definitions and we can proceed with the collapse result of interest.

Theorem 2.2.1
If $\mathsf{PostBQP} = \mathsf{PostBPP}$ *then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ *and the polynomial hierarchy collapses to the second level.*

Proof. Assume $\mathsf{PostBQP} = \mathsf{PostBPP}$. By Toda $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$, by Aaronson [19] $\mathsf{PostBQP} = \mathsf{PP}$ and by $\mathsf{PostBPP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ [20, 28]. This gives:

$$\mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}} \text{ and } \mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}} \subseteq \mathsf{P}^{\mathsf{BPP}^{\mathsf{NP}}} = \mathsf{BPP}^{\mathsf{NP}} = \mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}.$$

Therefore $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PH}$.

At this point we recall an interesting lemma from Toda [25] to continue.

$$\mathsf{PP}^{\mathsf{PH}} \subseteq \mathsf{BP} \cdot \mathsf{PP}$$

Clearly $\mathsf{PH} \subseteq \mathsf{PP}^{\mathsf{PH}}$ and it can be shown that if $\mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then $\mathsf{BP} \cdot \mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}$ (a full proof follows in lemma 2.2.1), therefore

$$\mathsf{PH} \subseteq \mathsf{BP} \cdot \mathsf{PP} \subseteq \mathsf{BPP}_{\|}^{\mathsf{NP}}.$$

Since $\mathsf{PH} = \mathsf{P}^{\#\mathsf{P}}$ and $\mathsf{BPP}_{\|}^{\mathsf{NP}} \subseteq \mathsf{PH}$ we can conclude $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$. We now have the condition $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ so by theorem 2.1.1 we get $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$, the final result. □

Lemma 2.2.1
If PP *is in* $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ *then so is* $\mathsf{BP} \cdot \mathsf{PP}$

Proof. Fix some $L \in \mathsf{BP} \cdot \mathsf{PP}$. By the definition, there exists $A \in \mathsf{PP}$ which defines L. Assume PP is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$, therefore A is in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. Therefore L is in $\mathsf{BP} \cdot \mathsf{BPP}_{\|}^{\mathsf{NP}}$.

Note that majority-vote amplification can be used on $\mathsf{BP} \cdot \mathsf{PP}$ and $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ to decrease the probability of failure to less than $1/6$.

If we combine the random coin flips of both the $\mathsf{BP} \cdot \mathsf{PP}$ and BPP parts of the algorithm we get a single failure probability below $1/3$, which fits the definition of $\mathsf{BPP}_{\|}^{\mathsf{NP}}$. □

Theorem 2.2.1 provides a quick a simple proof of Theorem 2.1.4 (the natural interpretation of our result).

Proof of Theorem 2.1.4. By definition $\#\mathsf{P} = \mathsf{ExactCount}$. O'Donnell and Say show $\mathsf{P}_{\|}^{\mathsf{ApproxCount}} = \mathsf{PostBPP}$ [20]. Therefore we can rewrite the antecedent of the Theorem as $\mathsf{P}_{\|}^{\#\mathsf{P}} = \mathsf{PostBPP}$.

Any problem in PP can be solved with one $\#\mathsf{P}$ query, which can obviously be done in one parallel round of oracle calls, so $\mathsf{PP} \subseteq \mathsf{P}_{\|}^{\#\mathsf{P}}$. This gives $\mathsf{PP} = \mathsf{PostBPP}$ (as we already know $\mathsf{PostBPP} \subseteq \mathsf{PP}$ and by Theorem 2.2.1 $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$. □

2.3 Improved Hardness of BosonSampling, IQP, and DQC1

In the original work on the hardness of Boson Sampling [17] two cases were considered: *the exact case*, where the probability that the algorithm would sample a given element must be at least multiplicatively close to the target distribution, and *the approximate case*, which allowed additive error in the total variation distance between the sampled and target distribution. For the exact case, classical simulation implied the polynomial hierarchy collapsed to the third level. For the approximate case, a collapse to the 3rd level was achieved, but subject to additional assumptions (we refer to these as additional assumptions henceforth). This separation between multiplicative and additive error also applies to the work on instantaneous quantum polynomial-time sampling (IQP) [18] and sampling from 1 clean qubit circuits (DQC1) [29], albeit with different additional assumptions. It is important to notice that realistic quantum computers are believed not to be able to achieve multiplicative error in sampling, but can achieve additive error [18, 30]. So it is the approximate case that distinguishes quantum from classical computation.

Fujii [21] et al strengthened the collapse for the exact case to $\mathsf{PH} = \mathsf{AM} \cap \mathsf{coAM}$ (a collapse to the second level of the polynomial hierarchy) for BosonSampling, DQC1 and IQP (amongst others). However, for fundamental reasons, the proof did not extend to the approximate case.

In this section, we show that efficient classical sampling of the physically relevant *approximate case* would also collapse the polynomial hierarchy to its second level ($\mathsf{ZPP}^{\mathsf{NP}}$), conditional on the existing additional assumptions. In this sense, our work provides the strongest known separation between classical and quantum computations for sampling and functional problems. While our result can also be applied to the exact case it is not an improvement on Fujii's result, as it is known $\mathsf{AM} \cap \mathsf{coAM} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$, and this application is therefore not relevant.

To apply Theorem 2.1.1 and show a second level PH collapse we require the condition $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}_{\|}$, however, the proofs contained in [17, 18, 29] only show $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$. Fortunately, each of the proofs can be easily modified to only use parallel oracle calls, giving $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}_{\|}$. Formally proving this fact would require reproducing each proof in detail and would make this chapter excessively long. Instead of completely rewriting these proofs we will instead notice that each proof only uses the NP oracle to approximately count some post-selected quantity with Stockmeyers algorithm [31]. In the next lemma, we show that this computation can

be done with parallel oracle calls. This will allow us to argue classical sampling of BosonSampling, IQP or DQC1 would imply $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ without formally reproducing each of the proofs.

Lemma 2.3.1 (Counting a postselected quantity)
Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a post selection criteria $h : \{0,1\}^n \to \{0,1\}$ let

$$p = \Pr_{x\in\{0,1\}^n}[f(x) = 1|h(x) = 1] = \frac{\sum_{x\in\{0,1\}^n} f(x)h(x)}{\sum_{x\in\{0,1\}^n} h(x)}.$$

Then for all $g \geq 1 + \frac{1}{poly(n)}$, there exists an $\mathsf{FBPP}_{\|}^{\mathsf{NP}^{\mathsf{f},\mathsf{h}}}$ machine that approximates p to within a multiplicative factor of g.

The proof of lemma 2.3.1 is quite trivial once it is realised that Stockmeyer's approximate counting theorem can be done with only parallel oracle calls, which we capture in the next lemma.

Lemma 2.3.2 (Approximate counting in parallel)
Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let

$$p = \Pr_{x\in\{0,1\}^n}[f(x) = 1] = \frac{1}{2^n}\sum_{x\in\{0,1\}^n} f(x).$$

Then for all $g \geq 1 + \frac{1}{poly(n)}$, there exists an $\mathsf{FBPP}_{\|}^{\mathsf{NP}^{\mathsf{f}}}$ machine that approximates p to within a multiplicative factor of g.

We will not provide all the steps of this lemma as it is a clear corollary of the version of the proof given by Valiant and Vazirani [32][4]. This theorem is also a consequence of the results of O'Donnel and Say [20], who show that approximate counting is in PostBPP, and therefore in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$.

We then proceed with the proof of lemma 2.3.1.

Proof of lemma 2.3.1. Fix a target error, g, from the assumption of the theorem there exists d such that $g > 1 + \frac{1}{n^d}$. By lemma 2.3.2 we can approximate $\sum_{x\in\{0,1\}^n} f(x)h(x)$ to a multiplicative factor of $g' = 1 + \frac{1}{3n^d}$ with high probability. Similarly we can approximate $\sum_{x\in\{0,1\}^n} h(x)$ to g' as well. Dividing the first sum by the second gives us p within a multiplicative factor of

$$g'^2 = 1 + \frac{2}{3n^d} + \frac{1}{9n^{2d}} \leq 1 + \frac{2}{3n^d} + \frac{1}{9n^d} = 1 + \frac{7}{9}\frac{1}{n^d} < 1 + \frac{1}{n^d}$$

[4]To see this, note that they can fix their random vectors at the start of their algorithm and check in parallel if 1, 2, up to n vectors are sufficient to empty the set.

with sufficiently high probability. Since each of these sums can be done in parallel and then divided for the final answer we can perform them in parallel. Given we only need to decrease the failure probability by a factor of 2 the final algorithm is in $\mathsf{FBPP}_{\|}^{\mathsf{NP}^f}$. □

We will first apply the above logic for BosonSampling results. Aaronson and Arkhipov use one post-selection step in lemma 42 in [17], they then perform one approximate counting step on a quantity derived from this post-selection step to prove their main theorem. The structure of this proof fits the format of lemma 2.3.1, therefore we can state a parallelised version of their main theorem.

Theorem 2.3.1 (Aaronson and Arkhipov with parallel calls)
Let $\mathcal{D}_A$ be the probability distribution sampled by a boson computer A. Suppose there exists a classical algorithm C that takes as input a description of A as well as an error bound ε, and that samples from a probability distribution $\mathcal{D}'_A$ such that $\|\mathcal{D}'_A - \mathcal{D}_A\| \leq \varepsilon$ in poly$(|A|, 1/\varepsilon)$ time. Then the $|GPE|^2_{\pm}$ problem is solvable in $\mathsf{BPP}_{\|}^{\mathsf{NP}}$.

The $|GPE|^2_{\pm}$ problem (defined in [17]) becomes #P complete given two conjectures: The **permanent-of-Gaussians Conjecture** (PGC) and the **Permanent Anti-Concentration Conjecture** (PACC). These conjectures capture the belief that the permanent of Gaussian matrices does not concentrate close to zero and is hard to estimate, further justification of these conjectures is available in the original paper [17]. These are the *additional assumptions* we referred to earlier.

Theorem 2.3.2
Assume PACC and PGC hold. If there exists a classical algorithm that can sample the distribution of a boson computer with additive error, then $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$ and the polynomial hierachy collapse to the second level.

Next, we discuss how the same strategy applies to the IQP case. The proof provided by Bremner, Montanaro and Shepherd uses just Stockmeyer counting to prove their collapse to $\mathsf{BPP}^{\mathsf{NP}}$ in their theorem 1. By lemma 2.3.2 all NP calls in their algorithm can be done in parallel and we can convert their theorem 1 to a $\mathsf{BPP}_{\|}^{\mathsf{NP}}$ result. For IQP we do not need to conjecture the concentration of some hard-problem. The result only rest on the conjectured average case hardness of approximately computing either the partition function of the Ising model *or* on a property of low-degree polynomials over finite fields (Conjectures 2 and 3 in [18]).

Theorem 2.3.3 (Improved IQP hardness [18])
Assume either above conjecture is true. If it is possible to classically sample from the output probability distribution of any IQP *circuit in polynomial time, up to an error of 1/192 in l_1 norm, then* $\mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\mathsf{NP}}$. *Hence the Polynomial Hierarchy would collapse to its second level.*

Finally, the same strategy can be applied to improve Morimae's result [29] on the hardness of the DQC1 model for sampling [21] as it again depends on Stockmeyer counting. The necessary conjecture for Morimae's is an assumption directly about the average case hardness of the one clean qubit model.

Theorem 2.3.4 (Improved one clean qubit hardness [29])
Assuming Morimae's conjecture, if there exists a classical algorithm which can output samples from any one clean qubit machine with at most 1/36 error in the l_1 norm then there is a $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm to solve any problem in* $\mathsf{P}^{\#\mathsf{P}}$. *Hence the Polynomial Hierarchy would collapse to its second level.*

It seems likely that other results will fit the structure we give here, although we provide only these three results.

We finish this section by noting that the above results are in SampBQP, so classical impossibility on any of these tasks would imply $\mathsf{SampBQP} \neq \mathsf{SampP}$, which would also imply a separation in the functional classes $\mathsf{FBQP} \neq \mathsf{FBPP}$ [14].

Theorem 2.3.5
If any of the sets of conjecture above hold and the polynomial hierarchy does not collapse to it's second level, then $\mathsf{FBQP} \neq \mathsf{FBPP}$ *and equivalently* $\mathsf{SampBQP} \neq \mathsf{SampP}$.

2.4 Conclusion and Open Problems

This chapter has shown that if $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}_{\|}^{\mathsf{NP}}$ then the polynomial hierarchy collapses to its second level. We have connected this result to approximate/exact counting and shown that it improves a number of results demonstrating the separation of quantum and classical computing. The natural next research question is 'how low can we go?'. Fujii et al [21] extended the result for the multiplicative error case to $\mathsf{AM} \cap \mathsf{coAM}$, perhaps this could be achieved. We have not used the fact that the $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm likely only needs two rounds of oracle calls. This could be an avenue to collapsing the hierarchy further.

This hierarchy collapse strengthens claims of quantum advantage but as these claims also rest on a number of additional assumptions (e.g. permanents-of-Gaussians conjectures), diminishing, removing or proving the other assumptions remain one of the central challenges of showing quantum advantage and proving $\mathsf{SampBQP} \neq \mathsf{SampP}$. Alternatively further work may reveal one of these assumptions to fall through, such as with XQUATH [30].

Our results offer other, more direct, extensions. Of particular interest is whether our main theorem relativises, like previous advantage results did [17]. Avoiding using the checkability of #P may be key to proving relativisation as this is the only step of our proof that did not relativise.

Another open question is how Theorem 2.1.1 may be used for other non-quantum purposes, perhaps where approximate and exact counting are being compared. Alternatively, a research line we did not pursue is extending theorem 2.1.1. Extensions could be to other checkable rsr languages, perhaps PSPACE or EXP (although these may only be adaptively-rsr [23]), or to showing the equality holds for other elements of the polynomial hierarchy ($\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\Sigma_i}_{\|} \stackrel{?}{\Longrightarrow} \mathsf{P}^{\#\mathsf{P}} = \mathsf{ZPP}^{\Sigma_i}$).

CHAPTER 3

On Bounded Advice Classes

3.1 Introduction

A common concept in complexity theory is 'advice', where a Turing machine is provided with an advice string alongside its input. This string can provide any conceivable advice on how to solve problems up to size n but must be the same advice string for all inputs of a given size. As this advice string could be anything, advice classes are often much more powerful than their unadvised counterparts, containing problems which are otherwise undecidable. This powerful advice string allows advice classes to act as useful models for problems with long/expensive preprocessing phases (modelled as advice) followed by faster/cheaper processing phases. However, in many of these situations, the advice is produced by a more powerful Turing machine, but not an infinitely more powerful Turing machine, as would be the case in standard advice classes. In such situations, it would be more appropriate to study a form of 'bounded advice', where we ask what a given Turing machine of some complexity can accomplish when given advice generated by a more powerful, but not unbounded, Turing machine. For many cases where we might want to use advice classes, this bounded advice captures a much tighter idea of what we are trying to model:

- Cryptography, when an adversary may be able to expend immense

resources if the information learnt allows them to break an encryption scheme for a much smaller cost during run-time [33, 34].

- Quantum computing, where the quantum computer is used to prepare an algorithm which will run classically [35–37], or where a classical machine learning algorithm attempts to replicate the quantum model just using data from the quantum model [38].

- Machine learning, where a large and expensive training run is used to compute a smaller number of weights for a machine learning model. Once these weights are computed the model is often comparatively quite cheap to run [39–41].

With bounded advice classes, we can address these scenarios, we can study how a cryptographic scheme performs against an attacker with 'only' an EXP generated cheat sheet, instead of an unbounded cheat sheet. Formalising bounded advice allows us to exactly extract these ideas, and to ask our key question: *When is advice generated by a given Turing machine useful?* In this context, we can informally say a certain complexity class is a useful advice generator to another complexity class when the former class can be used to create advice strings that allow the latter class to recognize strictly more languages.

In Section 3.2 we give common notation and define the class of problems solvable by advice generated by a certain complexity class, e.g. $\mathsf{P/poly}^{\mathsf{EXP}}$ is the class of problems solvable by a polynomial-time Turing machine with access to an exponential-time machine. We also review closely connected work on the complexity of advice functions [27, 39, 42–45]. This previous work typically studies the functional complexity of generating the advice for a given problem in $\mathsf{P/poly}$. While our results allow us to push the state of the art in this research direction (we are able to improve the strongest result), we primarily focus on a closely related, but different question: 'For a given complexity of advice generator, what problems can be solved?'.

In Section 3.3 we prove a connection between bounded advice and unary languages. Namely:

$$\mathsf{P/poly}^{\mathsf{B}} = \mathsf{P/poly}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})} = \mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})},$$

where $\mathsf{Un}(\mathsf{B})$ denotes the set of all unary languages in a class B.

This connection allows us to directly produce results on when given complexity classes generate useful advice in Section 3.4.

- $\mathsf{P} \subsetneq \mathsf{P/poly}^{\mathsf{EXP}}$

- $\mathsf{P} \subsetneq \mathsf{P/poly}^{\mathsf{NP}} \iff \mathsf{EXP} \neq \mathsf{EXP}^{\mathsf{NP}}$
- $\mathsf{P} \subsetneq \mathsf{P/poly}^{\mathsf{PSPACE}} \iff \mathsf{EXP} \neq \mathsf{EXPSPACE}$

We connect our framework to existing quantum classes, $\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{P/poly}^{\mathsf{Un}\left(\mathsf{ZPP}^{\mathsf{NP}^{\mathsf{BQP}}}\right)}$[1] and with randomised advice $\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{P/rpoly}^{\mathsf{BQP}}$ in Section 3.4.2. This section also connects bounded quantum advice (where the advice itself is a quantum state) into our framework, proving the bounded-advice equivalent of the well known result, $\mathsf{BQP/qpoly} \subseteq \mathsf{QMA/poly}$ [46]:

$$\mathsf{BQP/qpoly}^{\mathsf{B}} \subseteq \mathsf{QMA/poly}^{\mathsf{ZPP}^{\mathsf{QMA}^{\mathsf{B}}}}.$$

3.2 Background

3.2.1 General notation and definitions

This work makes use of notation or definitions that, while standard, may only be known to readers of specific backgrounds. To aid with readibility by a wide audience we provide a short list of notation, definitions of common complexity classes should be checked in the complexity zoo [47].

- **The symbol '#':** It is often useful to join two inputs together to pass them both to a Turing machine, e.g. if we want to calculate $x + y$ for $x = 010$ and $y = 11$ we will need to pass both x and y, but as both are written in binary a simple concatenation makes it unclear where x ends and y begins, $xy = 01011$. To solve this problem we introduce a special symbol to our alphabet, #, which will be used to simply demarcate where two strings meet, e.g. $x\#y = 010\#11$
- **Prefixes:** We say x *is a length* n *prefix of* y if x is length n and the first n letters of y are equal to x, e.g. $x = 100$ is a length 3 prefix of $y = 100010$.
- **Turing machine standardisation:** Unless otherwise specified a Turing machine is assumed to be a polynomial time deterministic Turing machine. Turing machine is often abbreviated to TM. Occasionally the TM will not be deterministic or not polynomial time, this will be specified or be clear from context (i.e. the previous line

[1] We will later define $\mathsf{BPP/samp}^{\mathsf{BQP}}$ as the class of problems which can be solved on a quantum computer or by a BPP machine with samples from the quantum computer

specified a non-deterministic Turing machine and the following line talks about 'this TM').

- **Unary:** A unary language is a language such that all elements $x \in L$ consist of a string of 1's, i.e. $\forall x \in L\ \exists m$ such that $x = 1^m$.

  The set of all unary languages is written TALLY, all unary languages in some complexity classes B could be written $\mathsf{TALLY} \cap \mathsf{B}$, but we find it is clearer to write $\mathsf{Un}(\mathsf{B}) := \mathsf{TALLY} \cap \mathsf{B}$.

- **Sparsity:** A sparse language is a language which has at most poly(n) elements of size n.

  The class of all sparse languages is written as SPARSE.

  The set of all sparse languages in a complexity class, B, can be written $\mathsf{SPARSE} \cap \mathsf{B}$ (the intersection of these two classes) but for notational clarity, we will write it as $\mathsf{SP}(\mathsf{B}) := \mathsf{SPARSE} \cap \mathsf{B}$. This significantly improves readability but may confuse readers familiar with an older form of relativisation notation: $\mathsf{B}(\mathsf{C}) := \mathsf{B}^{\mathsf{C}}$.

- **Which exponential time? E vs EXP.** There are two standard, but non-equivalent, definitions of exponential time decision problems: E, which is equal to $\mathsf{DTIME}(2^{\mathsf{O}(\mathsf{n})})$, and EXP, which is equal to $\bigcup_{\text{all polynomials } p} \mathsf{DTIME}(2^{\mathsf{p}(\mathsf{n})})$. This work will make it clear when our results apply to only one of these definitions or to both. Older works, some cited here, may not conform to this nomenclature.

- $\boldsymbol{L(x)}$**:** For a language L the function $L(x)$ is equal to 1 if $x \in L$ and 0 otherwise.

- $\mathbf{1^n}$**:** 1^n denotes a string of ones of length n. $1^4 = 1111$.

- $\mathsf{B} - \mathsf{C}$. For two complexity classes, B and C, the class $\mathsf{B} - \mathsf{C}$ is the set of all the languages in B that are not also in C. This notation comes from the definition of complexity classes as sets.

3.2.2 Oracle-machines and double oracles

The standard definition of an oracular complexity class is given as follows.

Definition 3.2.1 (Oracular classes)
The complexity class of problems solvable by an algorithm in B *with access*

to an oracle for a language L is B^L. This extends to define an oracular complexity class, C by taking the union:

$$\mathsf{B}^{\mathsf{C}} := \bigcup_{L \in \mathsf{C}} \mathsf{B}^L$$

Sometimes a single Turing machine will have to make Oracle calls to two different languages, while this can be defined within the existing framework by creating a third language that can answer oracular calls to either language, we find it much simpler to define a double oracle machine.

Definition 3.2.2 (Double oracles)
For complexity classes B, C and D, a double oracle machine is a Turing machine, T, with oracular access to two problems, L_0, and L_1. The complexity class is:

$$\mathsf{D}^{\mathsf{B},\mathsf{C}} = \bigcup_{L_{\mathsf{B}} \in \mathsf{B}, L_{\mathsf{C}} \in \mathsf{C}} \mathsf{D}^{L_{\mathsf{B}}, L_{\mathsf{C}}}$$

In previous works the notation $L_{\mathsf{B}} \oplus L_{\mathsf{C}}$ is used for this purpose. While logically equivalent we find the comma notation to be much clearer, additionally it makes sparsity more apparent as either if either B or C (but not both) are sparse their set-addition may not be.

3.2.3 Bounded and unbounded advice classes

We can now move onto the meat of our definitions: advice classes.

Definition 3.2.3 (Advice classes)
A language L is in the advice class of B, B/poly, if there exists an integer, d, and a set of advice strings, $a = \{a_i : |a_i| < di^d\}_{i \in \mathbb{N}}$, such that the language

$$L' = \{x \# a_{|x|} : x \in L\}$$

is in B.

At this point the extension of this definition to bounded advice seems obvious, we just have to specify what it means for a complexity class to 'generate' a piece of advice. But it is not clear how a decision algorithm (which outputs a bit) 'in' C generates a piece of advice (a string of symbols). Fortunately, a lot of heavy lifting has already been done with the definition of 'transducers' [48], which, unlike the standard definition of a Turing machine, gives the entire final state of the tape as the output, where a regular Turing machine only outputs 'reject' or 'accept'.

Unfortunately, on closer inspection, transducers do not have the properties we want; consider polynomial-sized advice generated by an exponential time machine, which polynomially-sized piece of the exponentially long tape should we take? Or a non-deterministic Turing machine, which path do we accept? What about for even more artificial classes, like SPARSE, which is defined without reference to Turing machines at all! Previous works [27, 39, 42–45] have dealt with these problems by using functional complexity classes. While well-defined, functional complexity classes can have very different properties from their decision-class counterparts [49], they also lack a clear notion of unary languages, which is of key importance to this work.

To combat these issues we choose a definition of bounded advice that uses the tape from a polynomial-time Turing machine with oracular access to another class. This is equivalent to using a FP^{B} advice generator for some oracle class B. This definition solves our problem of defining the output tape but inherently transforms the advice generator into a P^{C} machine, which makes it difficult to study classes for which $\mathsf{P}^{\mathsf{C}} \neq \mathsf{C}$.

Definition 3.2.4 (Polynomial-sized bounded advice classes)
For arbitrary complexity classes, B *and* C*, a language L is in* $\mathsf{B}/\mathsf{poly}^{\mathsf{C}}$ *if there is an infinite set of advice strings, $\{a_n\}_{n\in\mathbb{N}}$, such that:*

- *(Advice Generation) There exists a deterministic polynomial-time 'advice generating' Turing machine, T, with oracular access to* C *which on input 1^n terminates in* poly(n) *time with a_n as the final state of its tape.*
- *(Advice Use) The language*

$$L' = \{x\#a_{|x|} : x \in L\}$$

is in B

The notion of *unbounded* advice classes extends beyond classes of the form B/poly; logarithmic advice is possible with B/log, quantum advice, consisting of quantum states is possible with $\mathsf{B}/\mathsf{qpoly}$, or randomised advice can be captured with $\mathsf{B}/\mathsf{rpoly}$ [49]. Similarly, different flavours of bounded advice are possible; $\mathsf{B}/\mathsf{exp}^{\mathsf{C}}$ is the class of problems solvable by B given advice generated by a Turing machine running for exponentially long, or a quantum advice generator $\mathsf{B}/\mathsf{qpoly}^{\mathsf{C}}$, or even a non-deterministic advice generator $\mathsf{B}/\mathsf{npoly}^{\mathsf{C}}$. Exploring these definitions may allow future papers to work around the 'P-oracle problem' (how to study advice generated by

a class C such that $\mathsf{P}^{\mathsf{C}} \neq \mathsf{C}$). For our purposes, we will only need one of the possible extensions of bounded advice.

Definition 3.2.5 ($\mathsf{BQP/qpoly}^{\mathsf{C}}$)
For a fixed gate set and initial state $|0\rangle$, a language L is in $\mathsf{BQP/qpoly}^{\mathsf{C}}$ if the following two criteria are true:

- *(Advice Generation) There exists a polynomial-sized uniform family of quantum circuits with oracular access to C, $\{Q_n\}_{n\in\mathbb{N}}$ generating a set of advice states, $\{|\phi_n\rangle = Q_n |0\rangle | n \in \mathbb{N}\}$.*

- *(BQP Advice Use) There exists a polynomial-time quantum algorithm A such that for all n, $A(x, |\phi_n\rangle)$ outputs $L(x)$ with probability more than $2/3$ for all x up to length n.*

3.2.4 Previous work

We are not the first to consider bounded advice classes although a seemingly minor difference in our framing brings us to a novel set of questions and results. Much of the previous work on bounded advice has centred on the question 'If a given class A is in $\mathsf{P/poly}$, what is the complexity of generating the advice string?'. This naturally leads to studying the complexity of the 'advice function', a function which prints the advice used by the $\mathsf{P/poly}$ algorithm. The strongest answer to this question was noticed by Köbler and Watanabe [27] as a corollary to a result by Bshouty et al. [45], showing that the advice function for any $\mathsf{A} \in \mathsf{P/poly}$ is in $\mathsf{FZPP}^{\mathsf{NP}^{\mathsf{A}}}$.

While our bounded advice classes are equivalent to studying the advice function, the reframing leads to a different set of questions. Instead of asking 'for a given problem, what is the advice function to solve this?' we ask 'for a given advice function, which problems can this solve?'. This naturally leads to our central question of 'when are certain advice classes useful?'. It also forces us to standardise our notation and classes, which proves useful to deriving more flexible results. Finally, instead of studying functional classes to specify the advice function, we choose to stay within decision classes. This allows us to use the well-known connection between advice classes and unary languages.

For clarity, we will restate the state of the art result by Bshouty/Köbler using our terminology.

Theorem 3.2.1 ([27, 45])
For any set $\mathsf{A} \in \mathsf{P/poly}$: $\mathsf{A} \in \mathsf{P/poly}^{\mathsf{ZPP}^{\mathsf{NP}^{\mathsf{A}}}}$

Reexamining the proof of [45] it is easy to see this generalises to B/poly. We can also use the connection to unary languages we derive later to produce a stronger version of Bshouty et al.'s theorem.

Theorem 3.2.2

For complexity class A, if $\mathsf{A} \subseteq \mathsf{C}$ *and* $\mathsf{A} \subseteq \mathsf{B}/\mathsf{poly}$ *then* $\mathsf{A} \subseteq \mathsf{B}/\mathsf{poly}^{\mathsf{Un}\left(\mathsf{ZPP}^{\mathsf{NP}^{\mathsf{B},\mathsf{C}}}\right)}$

Many of our other results will rely on the well-known connection between advice classes and unary/sparse languages. As most research in advice functions has been functional languages this connection has scarcely been used. The only connection we are aware of is to a 1985 result by Ko and Schöning [43], showing that all sparse sets in Σ_i^p are also in P/poly with an advice function in $\mathsf{F}\Delta_{i+1}^p$.

3.3 Connection to sparse and unary languages

It is well known that any language in P/poly is Turing reducible to a unary language. Here we prove an analogous result, that any language in $\mathsf{P}/\mathsf{poly}^{\mathsf{B}}$ is Turing reducible to a unary language in P^{B}. We show this inclusion is tight, i.e. the set $\mathsf{P}/\mathsf{poly}^{\mathsf{B}}$ is exactly the set of languages which are Turing reducible to a unary language in P^{B}. We further show that all languages that are Turing reducible to sparse languages in P^{B} (i.e. in $\mathsf{P}^{\mathsf{SP}(\mathsf{P}^{\mathsf{B}})}$) are also in $\mathsf{P}/\mathsf{poly}^{\mathsf{NP}^{\mathsf{B}}}$.

To prove the main results we will need the following lemmas:

Lemma 3.3.1
For any complexity class B*:*

$$\mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})} \subseteq \mathsf{P}/\mathsf{poly}^{\mathsf{B}}.$$

Proof. As $\mathsf{Un}(\mathsf{P}^{\mathsf{B}})$ has at most n elements up to length n, the advice can simply be an n length string, a, whose m'th element is 1 iff $1^m \in L$. Obviously, this string can be generated by n queries to an P^{B} language. The advice-receiving Turing machine can then use this advice to simulate oracle calls to the P^{B} language. □

Lemma 3.3.2
For any complexity classes B *and* C*,*

$$\mathsf{C}/\mathsf{poly}^{\mathsf{B}} \subseteq \mathsf{P}^{\mathsf{C},\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$$

Proof. Take any $L \in \mathsf{C/poly}^{\mathsf{B}}$, let a_n be the advice string generated by the advice generator, and let $p(n)$ be the polynomial which bounds the length of the advice string ($|a_n| \leq p(n)$), W.L.O.G. we may assume the advice is a bit string of exactly $p(n)$ bits. We will now describe an oracle in $\mathsf{Un}(\mathsf{P}^{\mathsf{B}})$ that can be used to generate a_n in polynomial time.

Define the language:

$$L_{advice} = \{1^{\sum_{k=1}^{n-1} p(k)} 1^m | \text{ the m'th bit of } a_n \text{ is 1 } \}$$

By the definition of bounded advice a_n can be generated in polynomial time with a B oracle, deciding if a particular bit is 1 or 0 is also polynomial time. Therefore L_{advice} is a unary language in P^{B}.

The language L_{advice} can be used to construct a_n one bit at a time. This advice string can then be passed to the C oracle to simulate the $\mathsf{C/poly}^{\mathsf{B}}$ algorithm, showing that the language L is in $\mathsf{P}^{\mathsf{C},\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$. □

Combining lemma 3.3.1 and lemma 3.3.2 we get the central theorem of this section:

Theorem 3.3.1
For any complexity class B*,*

$$\mathsf{P/poly}^{\mathsf{B}} = \mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$$

A simple corollary, $\mathsf{P/poly}^{\mathsf{B}} = \mathsf{P/poly}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$, provides us with a key insight: all bounded advice is unary advice.

The result of Theorem 3.3.1 also holds for advice receivers other than P. The choice of fixing P as the advice receiver was merely to ensure Turing reductions. Following the steps of the proof for other classes provides the following corollary.

Corollary 3.3.2

$$\mathsf{BPP/poly}^{\mathsf{B}} = \mathsf{BPP}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$$

$$\mathsf{NP} \cap \mathsf{coNP/poly}^{\mathsf{B}} = (\mathsf{NP} \cap \mathsf{coNP})^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$$

$$\mathsf{BQP/poly}^{\mathsf{B}} = \mathsf{BQP}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$$

While the connection to unary languages is tighter, bounded advice can also be connected to sparsity, finding that sparse languages always sit inside some bounded advice class (Theorem 3.3.3), and bounded advice classes are contained in sparse languages (Corollary 3.3.4). Theorem 3.3.3

is similar to the work of Ko and Schöning [43] who show that all sparse sets in Σ_i^p are also in $\mathsf{P/poly}$ with an advice function in $\mathsf{F}\Delta_{i+1}^p$. Our result differs as it is both wider-reaching (applying to classes outside the polynomial hierarchy) and showing inclusions in both directions (advice inside sparsity, and sparsity inside advice).

Theorem 3.3.3
For any complexity class B*,*

$$\mathsf{P}^{\mathsf{SP}(\mathsf{P}^{\mathsf{B}})} \subseteq \mathsf{P/poly}^{\mathsf{SP}(\mathsf{NP}^{\mathsf{B}})}$$

Proof. Let K be a language in $\mathsf{P}^{\mathsf{SP}(\mathsf{P}^{\mathsf{B}})}$. Then there exists a sparse language, L in P^{B} such that $K \in P^L$. As L is sparse there are at most polynomially many strings $x \in L$ of any particular length, n. Our strategy will be to find all of these strings with the advice generating TM, to create a list, $A_n = \{x : x \in L, |x| \leq n\}$, and pass A_n as advice. The advice using TM can then simulate an oracle call to L by simply checking if x is on the list.

It is possible to generate A_n in polynomial time with access to an NP^{B} oracle, with access to the (sparse) language:

$$L'_{sparse} = \{1^n \# p : \exists x \text{ with } x \in L, \text{ and } p \text{ is a prefix of } x\}.$$

As L is in P^{B} existence of an $x \in L$ is in NP^{B}.

We can use L'_{sparse} to find a complete list of strings by beginning with an empty list, $A = \{\}$. We begin by using algorithm 1 starting from an empty string to find an initial element, $x \in L$, and put this element in A. If we try to nïavely reuse algorithm 1 to find more elements it may not return a new value of $x \in L$. We must force the algorithm to return a new value of x. We can solve this problem by beginning the search from some particular prefix, instead of the empty string.

Suppose our list, A, has m elements in it. This defines a partially explored trie, as shown in figure 3.1. At each node there are two children, one child must be an explored path that leads to at least one previously found element of $x \in A$, the other child may be unexplored. By beginning the search with a prefix from one of the unexplored children (highlighted in step b of figure 3.1) we will find any unfound elements beginning with that prefix. If a new element is found we add it to A and search its unexplored prefixes. If no new elements are found after all prefixes have been explored then we conclude the list A is complete. This list is our advice.

As A_n has at most polynomial elements and the tree is polynomially deep, finding each element takes at most polynomial time. Creating the

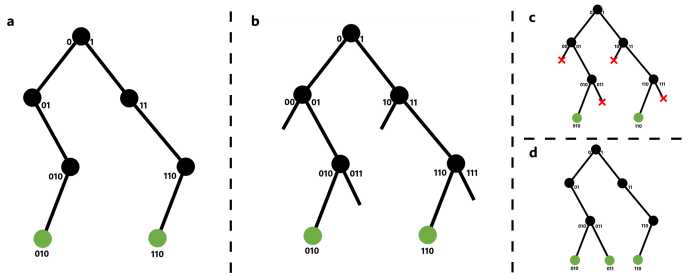


Figure 3.1: Step of the procedure for finding all the elements of a sparse language using L'. We begin with a possibly incomplete list, A, consisting of elements of L below to some given size, n. In step **a.** we represent A with a prefix tree, each layer representing a possible prefix of the string. If a string is a prefix for an element in A its branch continues to a green node. **b.** We add the possible unexplored children to the tree. We then use algorithm 1 starting from each unexplored child prefixes to find possible new elements of A. Either one is found and added to A (**d.**) or none are found, implying there are none left to find and A is complete (**c.**).

advice string A therefore takes polynomial time with an NP^{B} oracle, as L'_{sparse} is also sparse, we have shown the result:

$$\mathsf{P}^{\mathsf{SP}(\mathsf{P}^{\mathsf{B}})} \subseteq \mathsf{P/poly}^{\mathsf{SP}(\mathsf{NP}^{\mathsf{B}})}.$$

□

Theorem 3.3.3 shows sparse languages are in a form of bounded advice, the converse is also possible as a simple corollary of Theorem 3.3.1 as $\mathsf{Un}(\mathsf{B}) \subseteq \mathsf{Sp}(\mathsf{B})$.

Corollary 3.3.4
$\mathsf{P/poly}^{\mathsf{B}} \subseteq \mathsf{P}^{\mathsf{SP}(\mathsf{P}^{\mathsf{B}})}$

3.4 When is bounded advice useful?

This section will address this chapter's fundamental question: *"When is advice generated by a given complexity class useful?"* i.e. when advice from one class increases the amount of problems that can be solved by another class. In the first subsection, we will characterise when some major complexity classes generate advice useful to a polynomial time machine (i.e. when $\mathsf{P/poly}^{\mathsf{A}}$ is not simply equal to P). In the second

Algorithm 1 Binary search variant to find a_n from L'_{sparse}

Input: An integer n, a starting prefix, p
Output: The string a'

```
a' ← p                                  ▷ Initializing a' to the given prefix
advice_complete ← False                 ▷ Initializing advice_complete to False
while advice_complete is False
    if 1^n#a'0 ∈ L'_sparse              ▷ Find a symbol to append to the advice string
        a' ← a'0
    else if 1^n#a'1 ∈ L'_sparse
        a' ← a'1
    else
        advice_complete ← True          ▷ If no next symbol can be found, then the string is complete
```


subsection, we look at when quantum complexity classes are useful advice generators, this allows us to study a class of proposed uses of quantum computing that prepare classical algorithms. We connect $\mathsf{BPP/rpoly}^{\mathsf{BQP}}$ to the class of languages that can be decided with samples from a quantum computer through $\mathsf{BPP/samp}$[38]. We derive a result connecting quantum bounded polynomial advice (i.e. a quantum state) to non-quantum bounded polynomial advice (a classical bitstring).

3.4.1 Useful classical advice

As shown in the previous section $\mathsf{P/poly}^{\mathsf{B}} = \mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{B}})}$, thus bounded advice is useful if and only if unary languages are useful oracles. Much is known about the conditions for the existence of various unary languages, thus, this connection opens a variety of choices for grounding the hardness of various bounded advice classes. We provide a number of these results for the most famous complexity classes.

Theorem 3.4.1
$\mathsf{P} \neq \mathsf{P/poly}^{\mathsf{EXP}}$

Theorem 3.4.2
$\mathsf{P} \neq \mathsf{P/poly}^{\mathsf{PSPACE}}$ *if and only if* $\mathsf{EXPSPACE} \neq \mathsf{EXP}$.

Theorem 3.4.3
$\mathsf{P} \neq \mathsf{P/poly}^{\mathsf{NP}}$ *if and only if* $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{EXP}$.

Theorem 3.4.4
$\mathsf{BPP} \neq \mathsf{BPP/poly}^{\mathsf{BQP}}$ *if and only if* $\mathsf{BPEXP} \neq \mathsf{BQEXP}$

Many other results in unary languages can be used to prove results in advice classes. Such as showing the polynomial hierarchy gives useful advice if the exponential hierarchy doesn't collapse. However, for brevity, we have provided only these 4.

Proof of Theorem 3.4.1. We prove this result via showing $\mathsf{P} \neq \mathsf{P}^{\mathsf{Un(EXP)}}$. By the time hierarchy theorem, there exists an L such that,

$$L \in \mathsf{DTIME}(2^{n^3}) - \mathsf{DTIME}(2^{n^2}).$$

From L we define the unary language:

$$L_{unary} = \{1^x : x \in L\}$$

L_{unary} cannot be decided in $o(2^{log(n)^2})$, a quasipolynomial, thus $\mathsf{L_{unary}} \notin \mathsf{P}$.

Given a string from L_{unary}, 1^x, calculating x takes x steps, therefore L_{unary} can be decided by a deterministic Turing machine in $O(2^{log(n)^3})$ time. As $\mathsf{DTIME}(2^{\mathsf{log(n)}^3}) \subseteq \mathsf{EXP}$, $L_{unary} \in \mathsf{EXP}$. Therefore we have demonstrated there exists a unary language in $\mathsf{EXP} - \mathsf{P}$, by theorem 3.3.1 this implies $\mathsf{P/poly}^{\mathsf{EXP}} \neq \mathsf{P}$ □

Proof of Theorem 3.4.2. By Theorem 3.3.1 $\mathsf{P/poly}^{\mathsf{PSPACE}} = \mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{PSPACE}})}$, as $\mathsf{P}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$ [47] we derive the equality: $\mathsf{P/poly}^{\mathsf{PSPACE}} = \mathsf{P}^{\mathsf{Un(PSPACE)}}$. If $\mathsf{EXPSPACE} = \mathsf{EXP}$ there are no unary languages in $\mathsf{PSPACE} - \mathsf{P}$ [50], proving the 'only if' direction. For the other direction we notice that all unary languages in PSPACE are in $\mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{PSPACE}})}$, therefore if $\mathsf{P}^{\mathsf{Un}(\mathsf{P}^{\mathsf{PSPACE}})} = \mathsf{P}$ there are no unary languages in PSPACE and $\mathsf{EXPSPACE} = \mathsf{EXP}$ [50]. □

Proof of Theorem 3.4.3. There are unary languages in $\mathsf{P}^{\mathsf{NP}} - \mathsf{P}$ if and only $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{EXP}$. Therefore $\mathsf{P/poly}^{\mathsf{NP}} \neq \mathsf{P}$ if and only if $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{EXP}$. □

Proof of Theorem 3.4.4. There are unary languages in $\mathsf{BQP} - \mathsf{BPP}$ if and only $\mathsf{BPEXP} \neq \mathsf{BQEXP}$, by a simple extension of the arguments in [50]. Therefore $\mathsf{P/poly}^{\mathsf{BQP}} \neq \mathsf{P}$ if and only if $\mathsf{BQEXP} \neq \mathsf{BPEXP}$. □

3.4.2 Useful Quantum advice

This subsection studies bounded quantum advice and when it is useful. First, we examine classical advice strings generated by quantum computers, this case contains many of the proposed algorithms to use quantum

computers to prepare algorithms for classical machines [35, 36, 38, 51]. One prominent example, using samples from some problem to produce an algorithm that can solve instances of the same problem, is formalised in the class $\mathsf{BPP/samp}$, if the samples are produced from a quantum computer, we denote it as the class $\mathsf{BPP/samp^{BQP}}$ [38]. While it may be expected that $\mathsf{BPP/samp^{BQP}}$ is contained in $\mathsf{P/poly^{BQP}}$ this neglects the randomness in the sample selection, and we will show it is instead contained in $\mathsf{P/rpoly^{BQP}}$. We also show a derandomised result, that $\mathsf{BPP/samp^{BQP}}$ is contained in $\mathsf{P/poly}^{\mathsf{Un}\left(\mathsf{ZPP}^{\mathsf{NP}^{\mathsf{BQP}}}\right)}$. Second, we study the properties of bounded advice when the advice is itself a quantum state: $\mathsf{BQP/qpoly^B}$, finding it is contained in a classical bounded advice state $\mathsf{QMA} \cap \mathsf{coQMA/poly}^{\mathsf{Un}(\mathsf{ZPP}^{\mathsf{QMA}^{\mathsf{B}}})}$.

Definition 3.4.1 ($\mathsf{BPP/samp}$ [38])
A language L is in $\mathsf{BPP/samp}$ if there exists probabilistic polynomial-time Turing machines M and D with the following properties: On input 1^n, D produces output distribution $\mathcal{D}_n$. M takes an input of size n along with 'samples' from L, $\mathcal{T} = \{(x_i, L(x_i))\}_{i=1}^{poly(n)}$, where x_i is sampled from $\mathcal{D}_n$. M outputs 1 with probability greater than $2/3$ if $x \in L$, and less than $1/3$ if $x \notin L$, where the probability taken is over the randomness in both sample selection and random coins.

The original definition of $\mathsf{BPP/samp}$ [38] is not explicit which domain the 2/3 failure probability applies to, it could be that for most sets of samples the machine M must function 'according to the rules of BPP' (for all points there is a 2/3 probability of failure over the coin flips), or it could be that for each point, a 2/3 probability of failure applies over both the set of samples and the set of coin flips. While the former appears to be a much tighter restriction (the majority of all sets work on all points for most sets of coin flips), fortunately, these two definitions are equivalent via a simple boosting-and-majority-vote argument. Similarly, it is clear that the use of BPP in the definition of $\mathsf{BPP/samp}$ is superfluous as if there is randomness over the samples, this randomness is sufficient to use as randomness to simulate coin flips. If a given $\mathsf{BPP/samp}$ algorithm requires m random coins, we can ask for extra samples and use the random inputs x as random coins[2].

Theorem 3.4.5

[2]The random distribution of samples, $\mathcal{D}_n$, might not be uniform, infact there is no requirement for it to be non-deterministic. Fortunately, we can always append $m/|x|$ uniform random samples on the end of our set of samples to use as random coin flips.

$\mathsf{BPP/samp} = \mathsf{P/samp}$

For our purposes, we are interested in exactly the restriction of $\mathsf{BPP/samp}$ to samples that can be produced by a particular machine (i.e. a quantum machine/BQP). We define $\mathsf{BPP/samp}^{\mathsf{B}}$ equivalently to $\mathsf{BPP/samp}$ but requiring that the labelling problem $(L(x))$ is in the complexity class B. Fortunately, this is simply $\mathsf{BPP/samp} \cap \mathsf{B}$.

Lemma 3.4.1
$\mathsf{BPP/samp}^{\mathsf{B}} = \mathsf{BPP/samp} \cap \mathsf{B}$

From this definition, the following result is immediately clear:

Theorem 3.4.6
$\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{BPP/rpoly}^{\mathsf{BQP}}$

As noted by Watrous [52] randomised advice can be used to give random coins to a probabilistic algorithm, which extends to bounded advice to show $\mathsf{BPP/rpoly}^{\mathsf{BQP}} = \mathsf{P/rpoly}^{\mathsf{BQP}}$. This connection improves Theorem 3.4.6 to the following corollary.

Corollary 3.4.7
$\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{P/rpoly}^{\mathsf{BQP}}$

As noted above, the definition of $\mathsf{BPP/samp}$ seems to require randomised advice. Fortunately, we can use Theorem 3.4.6 to derandomise this advice, connecting $\mathsf{BPP/samp}^{\mathsf{BQP}}$ to a deterministic bounded advice class.

Theorem 3.4.8

$$\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{P/poly}^{\mathsf{Un}\left(\mathsf{ZPP}^{\mathsf{NP}^{\mathsf{BQP}}}\right)}$$

Proof. As $\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{BQP}$ and $\mathsf{BPP/samp}^{\mathsf{BQP}} \subseteq \mathsf{P/poly}$, applying Corollary 3.2.2 produces the desired result. □

Similar these techniques bound other 'quantum preparation classes', such as $\mathrm{CSIM}_{\mathrm{QE}}$ [51].

We can now turn our attention to quantum states given as advice, the standard equivalent result in unbounded advice classes for connecting quantum advice to advice is $\mathsf{BQP/qpoly} \subseteq \mathsf{QMA} \cap \mathsf{coQMA/poly}$ [46], we show a close analogue exists for bounded advice classes:

Theorem 3.4.9

$$\mathsf{BQP/qpoly}^{\mathsf{A}} \subseteq \mathsf{QMA} \cap \mathsf{coQMA/poly}^{\mathsf{ZPP}^{\mathsf{QMA}^{\mathsf{A}}}}$$

Proof. Aaronson [46] and Drucker showed:

$$\mathsf{BQP/qpoly} \subseteq \mathsf{QMA} \cap \mathsf{coQMA/poly}$$

As $\mathsf{BQP/qpoly}^{\mathsf{A}} \subseteq \mathsf{BQP}^{\mathsf{A}}$ we can apply Theorem 3.2.1 to derive:

$$\mathsf{BQP/qpoly}^{\mathsf{A}} \subseteq \mathsf{QMA} \cap \mathsf{coQMA/poly}^{\mathsf{Un}(\mathsf{ZPP}^{\mathsf{QMA}^{\mathsf{QMA}\cap\mathsf{coQMA},\mathsf{BQP}^{\mathsf{A}}}})}$$

This result can be significantly simplified, first, we note that $\mathsf{QMA} \cap \mathsf{coQMA}$ is low for QMA

$$\mathsf{QMA}^{\mathsf{QMA}\cap\mathsf{coQMA},\mathsf{BQP}^{\mathsf{A}}} \subseteq \mathsf{QMA}^{\mathsf{BQP}^{\mathsf{A}}}$$

Then, we use $\mathsf{QMA}^{\mathsf{BQP}^{\mathsf{A}}} = \mathsf{QMA}^{\mathsf{A}}$ to derive the result

$$\mathsf{BQP/qpoly}^{\mathsf{A}} \subseteq \mathsf{QMA} \cap \mathsf{coQMA/poly}^{\mathsf{Un}(\mathsf{ZPP}^{\mathsf{QMA}^{\mathsf{A}}})}$$

□

CHAPTER 4

High Dimensional Quantum Machine Learning With Small Quantum Computers

4.1 Introduction

Quantum machine learning is often listed as one of the most promising applications of a near term quantum computer [53], with important early successes in a range of problems, from classification [54, 55] to generative modelling [56]. However, the broader roll out of these methods to real world problems is tempered, in part, by the limited size of quantum computers. Among other limitations, current quantum computers lack enough qubits to run large circuits. Some "circuit partitioning schemes" [5, 12, 57] have been proposed to simulate larger circuits on smaller devices by partitioning the full circuit into a set of smaller circuits (see figure 4.1). However, the exponential number of circuits needed by these schemes is completely intractable for most applications, with billions of sub-circuit evaluations required for even modest quantum machine learning instances.

In this work we examine the necessity of each subcircuit in producing an approximation of some partitioned circuit, presenting reasoning that a smaller amount of circuits could be sufficient in some cases. We then use this as inspiration for a new machine learning technique, which reconciles the need for larger circuit instances with affordable runtimes. Our new

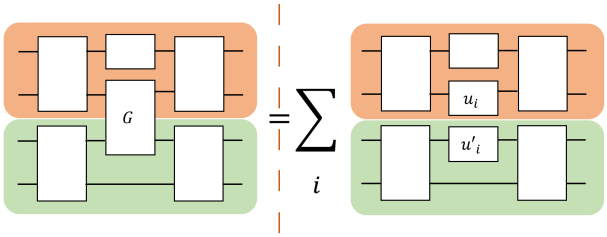


Figure 4.1: The fundamental notion of circuit partitioning. A potential partition (peach coloured and mint coloured) exists but is joined by a 2-qubit gate, G. By expressing G as a sum of single qubit unitaries u_i and u'_i we can simulate the ouput of the large circuit by only running circuits on either element of the partition (which requires a smaller quantum computer).

technique takes the same form as a given generic machine learning architecture that has been partitioned using the aforementioned techniques but with vastly fewer terms.

We develop the basic theory behind this technique in Section 4.3, consider its generalisation error in Section 4.4 and test it experimentally in Section 4.6 on instances of handwritten digit recognition using a 64 qubit ansatz with access to only a simulated 8 qubit computer (without use of excessive dimensionality reduction, such as dimensional principal component analysis). We also include an experiment testing the model's ability to replicate the output of larger unpartitioned circuits. Error analysis and the specifics of an evaluation and training schemes are presented in Section 4.5.

4.2 Related work

We are not the first to consider how the partitioning schemes [5, 12, 57] could be made more efficient, whereas other research lines have focused on minimising the computational cost of applying the *exact* partitioning schemes (e.g. by minimising the number of gates cut), we focus on shrinking the number of subcircuits to *approximate* the output. As such many of the techniques in this section can be composed with our method to create an even more efficient scheme.

In [58] an automated cutting procedure is applied to [57] to produce the minimum number of subcircuits needed. Similarly [59] uses maximum likelihood fragment tomography to improve both the cutting process and the reconstruction of output states. Other authors have considered how

the set of subcircuits could be run more effectively by utilising distributed computational resources [60, 61].

Using partitioning schemes produces an additional benefit: reduced noise, stemming from the smaller circuit size [58, 62], this can be the motivation for cut selection, even when the full circuit would "fit" on a quantum machine [63]. This noise reduction is similar to the increased accuracy we may be able to provide to gradients in our model. The potential link between this, and the avoidance of barren plateaus in our model we will discuss in Section 4.5.

After developing our technique we will demonstrate its use on high dimensionality data, specifically handwritten digit recognition. This problem has been tackled before with quantum hardware. In [64], dimensionality reduction techniques (such as principal component analysis) are used to reduce the dimensionality of the digits to a feature vector small enough to fit on their 8 qubit machine. Similarly, in [65] handwritten digits are classified on an 8 qubit machine, in this instance the size of the data is not reduced, the full data is carefully encoded into the quantum computer, first with amplitude encoding, and then by using 11 layers of parameterised gates. Our approach is fundamentally different from either of these. We use the same sized data (8×8 pixels) but do not apply dimensionality reduction as in [64], or reuse qubits for multiple data points as in [65]. We follow a simple encoding: giving each pixel its own qubit, which we can achieve as we are approximating a 64 qubit machine, while only using an 8 qubit machine.

Other works have addressed high dimensionality data by pushing the limit of the size of quantum machine learning models on current devices [66]. Some have employed quantum circuits as components of some larger algorithm to tackle bigger problems: [67] recursively applies PQCs (parameterised quantum circuits) to the outputs of PQCs and [68] uses quantum circuits as part of a hybrid tensor network but both do not address the task of partitioning a larger circuit and running it efficiently as we do here.

4.3 Model Motivation and Specification

In this section we introduce parameterised quantum circuits (PQCs), a popular concept in quantum machine learning; and circuit partitioning, a method of evaluating quantum circuits that requires a number of qubits greater than what is accessible. By applying these circuit partitioning schemes to PQCs we can produce a more powerful machine learning model than the smaller device naively allows, at the cost of unreasonable runtime.

We then go on to develop a novel QML method which intuitively may be as useful requiring only a fraction of the runtime.

4.3.1 Parameterised Quantum Circuits

Parameterised quantum circuits (PQCs) are a varied and promising method for quantum machine learning. In general they consist of some set of circuits, $\mathbb{U}$, parameterised by a weight vector, $\boldsymbol{\theta}$. In the most common forms the input datum, $\boldsymbol{x}$, also parameterises gates in the circuit. The set can be indexed as $\mathbb{U} = \{U(\boldsymbol{x}; \boldsymbol{\theta})\}$. These circuits yield functions when we specify an initial state, $|\phi\rangle$, and an observable, M:

$$f_{\boldsymbol{\theta},M,|\phi\rangle}(\mathbf{x}) = \langle\phi| U^\dagger(\boldsymbol{\theta}, \boldsymbol{x}) M U(\boldsymbol{\theta}, \boldsymbol{x}) |\phi\rangle \tag{4.1}$$

We can assume $|\phi\rangle$ is some fiducial state, such as $|0\rangle^{\otimes n}$ in the computational basis, without loss of generality.

As each setting of $\boldsymbol{\theta}$ defines a (not necessarily unique) function, $f_{\boldsymbol{\theta},M,|\phi\rangle}$, the set of unitaries defines a set of functions. We call this set the *hypothesis class*, to coincide with the common usage in machine learning. PQCs have been studied in other contexts, such as quantum chemistry or condensed matter physics [69], and although it is likely our approaches might generalise to these areas, in this work we focus on its application to machine learning.

Definition 4.3.1 (PQC hypothesis class)
The hypothesis class generated by the family of parameterised quantum circuits $\mathbb{U}$ together with an observable M is given by

$$\mathcal{F}_{\mathbb{U},M} = \Big\{ f_{\boldsymbol{\theta}}(\boldsymbol{x}) : f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle 0| U^\dagger(\boldsymbol{\theta}, \boldsymbol{x}) M U(\boldsymbol{\theta}, \boldsymbol{x}) |0\rangle : \boldsymbol{\theta} \in [0, 2\pi)^{N_{\mathrm{PS}}} \Big\},$$

where N_{PS} is the number of parameters in the model.

These PQCs have proven popular, but the implementation of PQCs is currently hindered by the NISQ machines they run on. Notably the limited number of qubits available limits the width (defined henceforth as number of qubits the circuit acts on) of the circuit that can be run. It is the central concern of this work to produce a model as useful as PQCs of width larger than what the available machines naively permit.

4.3.2 Circuit Partitioning

In [5, 12, 57] the authors propose methods to simulate large quantum circuits on smaller quantum machines by partitioning the circuit into smaller disconnected blocks. In this section we will introduce and then employ these methods on PQCs to decrease the size of quantum computer needed. In our work the decompositions are based on the result of [12] however extension to the results of [5, 57] are also possible. We present only the approach of [12] as they are, for our purposes, very similar.

Consider a partition of the N qubits into blocks, $\{B_i\}_i$, where $B_i \subset [N]$ ($[N] := \{1, 2, \ldots, N\}$) such that $\bigcup_i B_i = [N]$ and $B_i \cap B_j = \emptyset \; \forall i \neq j$ (i.e. each qubit is in one and only one block of the partition). We use the fact that any unitary matrix can be decomposed into a sum of weighted tensor products of single qubit unitaries. In [12] this fact is used to decompose any particular 2-qubit gate into a gate of the form:

$$U = \sum \alpha_i u_i \otimes u_i' \tag{4.2}$$

for some complex α_i such that $\sum |\alpha_i|^2 = 1$ and for 2 dimensional unitaries, u_i and u_i'. The number of terms of the sum needed for any particular gate is given by its Schmidt number [70], generically this number is 4 for 2-qubit entangling gates but for some important cases (including the CNOT and controlled-Z) only 2 terms are needed. For example, we can decompose the Controlled-Z gate into single qubit gates as:

$$\text{Controlled} - \text{Z} = \frac{1}{1+i}\left(S \otimes S + iS^\dagger \otimes S^\dagger\right) \tag{4.3}$$

where S is the phase gate [1]. The identity (4.2) allows us to rewrite any particular 2-qubit gate as the sum of products of single qubit operators. Applying this method to every 2-qubit gate connecting two blocks of the partition decomposes the full unitary into a sum of tensor products of unitaries which individually act only on each block of the partition (figure 4.1).

An example is useful in illustrating this point, suppose we are given a unitary W which consists of two disconnected blocks apart from one 2-qubit gate, G, connecting the otherwise disjoint blocks, top and bottom (as in figure 4.1):

$$W = (U_{\text{top}} \otimes U_{\text{bot}})G(V_{\text{top}} \otimes V_{\text{bot}}) \tag{4.4}$$

We can decompose this 2-qubit gate as $G = \sum_i \alpha_i u_i \otimes u_i'$. The full unitary

can thus be written as:

$$\begin{aligned} W =& (U_{\text{top}} \otimes U_{\text{bot}})(\sum_i \alpha_i u_i \otimes u_i')(V_{\text{top}} \otimes V_{\text{bot}}) \\ =& \sum_i \alpha_i (U_{\text{top}} \otimes U_{\text{bot}})(u_i \otimes u_i')(V_{\text{top}} \otimes V_{\text{bot}}) \\ =& \sum_i \alpha_i (U_{\text{top}} u_i V_{\text{top}}) \otimes (U_{\text{bot}} u_i' V_{\text{bot}}) \end{aligned} \tag{4.5}$$

Suppose the initial state is $|0\rangle^{\otimes n}$ (which we will simply refer to as $|0\rangle$) and the measurement is the projection, $|0\rangle\langle 0|$ (using the previous notation for $|0\rangle^{\otimes n}$), we then have that

$$\begin{aligned} \langle 0|W|0\rangle =& \\ & \langle 0| \sum_i \alpha_i (U_{\text{top}} u_i V_{\text{top}}) \otimes (U_{\text{bot}} u_i' V_{\text{bot}}) |0\rangle \\ =& \sum_i \alpha_i \langle 0| (U_{\text{top}} u_i V_{\text{top}}) |0\rangle \langle 0| (U_{\text{bot}} u_i' V_{\text{bot}}) |0\rangle \end{aligned}$$

and the expectation value is given by:

$$\begin{aligned} \Big| \langle 0|W|0\rangle \Big|^2 =& \\ & \sum_{i,j} \bar{\alpha_i} \alpha_j \langle 0| (U_{\text{top}} u_i V_{\text{top}})^\dagger |0\rangle \langle 0| (U_{\text{top}} u_j V_{\text{top}}) |0\rangle \\ & \cdot \langle 0| (U_{\text{bot}} u_i' V_{\text{bot}})^\dagger |0\rangle \langle 0| (U_{\text{bot}} u_j' V_{\text{bot}} |0\rangle . \end{aligned}$$

which is the product of inner products local to either element of the partition. This allows us to evaluate each smaller inner product individually and then combine them in a product and sum to replicate the expectation value of the full circuit. Depending on the observable it may be preferable to calculate the expectation value (i.e. the previous equation) or to calculate the inner product presented in the equation before and then square the answer to calculate the expectation value.

These results provide us with a clear path to solve the central goal of this chapter thus far, "How to fit a larger model on a smaller machine". It is simply a matter of specifying a large PQC, then deciding on a partition $\{B_i\}_i$ that separates its initial state and measurement nicely. This partition defines a set of closely related circuits that differ only by the replacement of 2-qubit gates with single qubit gates. The next theorem encapsulates

the partitioning of PQCs into a set of a set of smaller subcircuits, and the recombination of them to recreate the result of the larger PQC.

Theorem 4.3.1 (Partitioned model)
For every function $f_{\boldsymbol{\theta}} \in \mathcal{F}_{\mathbb{U},M}$ and qubit partition $\{B_k\}_{k\in[K]}$ with observable $M = \bigotimes_{k\in[K]} M_k$, there exists a set of coefficients $\{c_i\}_{i\in[T]}$ and unitaries $\tilde{\mathbb{U}} = \{U^{i,k}(\boldsymbol{\theta},\boldsymbol{x}), U'^{i,k}(\boldsymbol{\theta},\boldsymbol{x})\}_{i\in[T],\ k\in[K]}$ (where each $U^{i,k}$ and $U'^{i,k}$ acts on n_k qubits) which can be combined in a function:

$$\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{T} c_i \prod_{k=1}^{K} \langle 0| U'^{i,k\dagger}(\boldsymbol{\theta},\boldsymbol{x}) M_k U^{i,k}(\boldsymbol{\theta},\boldsymbol{x}) |0\rangle , \tag{4.6}$$

such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \tilde{f}_{\boldsymbol{\theta},M}(\boldsymbol{x})$ for every $\boldsymbol{\theta},\boldsymbol{x}$. For arbitrary gates the number of terms T grows as 16^r, where r is the number of gates across the partition, but for cut gates with known Schmidt number T is the product of the Schmidt number squared of each cut gate.

Remarks
In many cases the same subcircuit (or its complex conjugate) appears multiple times in equation 4.6. By storing its value in classical memory the total number of circuit evaluations can be brought down to 6^r where r is the number of gates across the partition (as mentioned in [12]). Bounding the number of evaluations needed for a given circuit is a task studied in [71] in the context of the scheme in [5].

It must also be noted that Equation 4.6 is composed of inner products, not expectation values, thus requiring 2 circuits to evaluate. Further details on this and the effects of error are considered in Section 4.5.

Mapping this theorem onto our example $K = 2$, the set of coefficients would be $\{\overline{\alpha}_i\alpha_j\}$ and the set of unitaries would be

$$\begin{aligned}&\big\{\, \{U_{\text{top}} u_i V_{\text{top}}, U_{\text{top}} u_j V_{\text{top}}, \}\,, \\ &\big\{U_{\text{bot}} u'_i V_{\text{bot}}, U_{\text{bot}} u'_j V_{\text{bot}}\big\}\,\big\}_{i,j}.\end{aligned}$$

This example also illustrates the similarity of terms in equation 4.6, for every "top" circuit is identical up to the replacement of u_i, u_j.

Theorem 4.3.1 is useful to our goal, we can fit any large PQC on a small machine, however we have paid a huge price in the need to run an exponential number of smaller circuits. Indeed given that most QML models are relatively densely connected and increasing depth can lead to improved performance, this exponential overhead in number of cut

connections is impractically costly. For example a 2-block division of the hardware efficient ansatz up to depth 6, such as those considered in [72] to solve a simple task would require over 2 billion distinct sub-circuit evaluations. This rough estimation motivates us to revise our goal to "how to fit a larger model on a smaller machine in an *acceptable* number of circuit evaluations".

If we are interested in exactly recreating the output of the circuit, this goal might be unattainable, unless we can find an exponential number of terms that perfectly cancel each other. There are fortunately several acceptable simplifications we can make to our goal. First, we are not concerned with the exact replication of the unitary. Since our input states are fixed to $|0\rangle$ we only care about the action of our recreated unitary on this state. Second, we may be content with approximate results, or perhaps even approximate results for *most* input data. Finally, our ultimate goal for a machine learning model, in many cases, is simply to output a binary classifier [73] (or another simpler discrete set of outputs) so we are not interested in keeping terms which contribute similarly as other terms in the final assignment of a class label.

With this in mind we will now define the subset partition model as the best possible approximation of the full result in theorem 4.3.1 keeping only L terms.

Definition 4.3.2 (Subset partition model)
For a partitioned model, $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$, with set of unitaries $\tilde{\mathbb{U}}$, we define the L-subset partition model as a function using the optimal L-sized subset of terms $I \subset \tilde{\mathbb{U}}$ given by:

$$\tilde{f}^{I}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i \in I} \lambda_i \prod_{k=1}^{K} \langle 0 | \, U'^{i,k\dagger}(\boldsymbol{\theta}, \boldsymbol{x}) M_k U^{i,k}(\boldsymbol{\theta}, \boldsymbol{x}) \, |0\rangle \, .$$

where we have introduced free parameters λ_i that can also be optimised over. In the above definition I and $\boldsymbol{\lambda}$ are optimised to produce the best approximation of $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$, for some given success metric.

Inner products can often involve complex numbers, therefore the output of the model may be complex. However, most machine learning scenarios require a real number, in these cases, we take the real part and discard the imaginary.

This model is a step towards our goal, if we are given the model it would be possible to run some approximation of the partitioned circuit on a small computer in acceptable time. However we lack the capacity

to chose the *optimal* set of "small circuits" I, in general choosing this set corresponds to a combinatorial optimisation problem. In the next section we will describe why this problem is challenging and produce a model that can work around it.

4.3.3 Reduced Partition Model

In the last section we tackled the problem of how to fit a large model on a smaller machine, but it required us to run an impractical number of circuits to achieve our goal, we introduced a model to get around this but it was impractical to optimise. We now consider a situation where we are given a runtime "budget", a hypothetical number of circuits, L, that we can afford to evaluate. Choosing which L circuits to evaluate from the set generated by the partitioning to perform optimally is an incredibly challenging combinatoric optimisation problem. This process is additionally complicated by the apparent need to use a quantum computer to assess if the circuits can be ignored. In this section we propose a relaxation of the problem: by parameterising the gates that replaced the 2-qubit gates in the circuit cutting process (henceforth called partition gates) such that all terms in the sum are identical up to these introduced parameters. The problem of optimising circuit selection becomes one of optimising the parameters of the partition gates.

The first step of this process is parameterising the gates introduced by the partition. There are many options for doing this, for example when cutting the controlled Z we get the decomposition in equation 4.3, replacing the 2-qubit gate on either qubit by S or $S^\dagger$. We then wish to create a new parameterised gate which takes a parameter ζ such that the parameterised gate is S when $\zeta = 0$ and $S^\dagger$ when $\zeta = 1$. Z^ζ composed with $S^\dagger$ is one choice. Defining ζ this way also allows us to extrapolate gates for $\zeta \notin \{0, 1\}$, creating a continuous parameter we can use for e.g. gradient descent.

We can use this partitioned-gate-parameterisation trick to replace the set $\{U^{i,k}(\boldsymbol{\theta}, \boldsymbol{x}), U'^{i,k}(\boldsymbol{\theta}, \boldsymbol{x})\}_{i,\,k}$, with a new set, $\{U^k(\boldsymbol{\theta}, \boldsymbol{x}, \zeta)\}_{k\in[K]}$, with just one parameterised unitary for each block of the partition, with different terms of the sum differentiated only by different parameters ζ.

Lemma 4.3.1
For every $\tilde{f}_{\boldsymbol{\theta}} \in \mathcal{F}^L_{\mathbb{U},M}$, *there exists a set of unitaries* $\{U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta})\}_{k\in[K]}$

and parameters $\boldsymbol{\lambda}, \boldsymbol{\zeta}$ *defining a function:*

$$\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{i\in[L]} \lambda_i \prod_{k\in[K]} \langle 0| U^{k\dagger}(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k+K}) |0\rangle , \tag{4.7}$$

such that $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x})$ *for every* $\boldsymbol{\theta}, \boldsymbol{x}$ *and for every observable that can be written as tensor product on the elements of the partition* $M = \bigotimes_k M_k$.

In this lemma we have used our new free parameter $\boldsymbol{\zeta}$ to parameterise the partition gates, the parameters needed for these partition gates could be calculated from the partitioning theorem or trained through gradient descent. As mentioned, the advantage of this step is that now all terms of the sum are equivalent to each other up to weight parameters λ_i and $\boldsymbol{\zeta}_{i,k}$. This is useful in the final model, where we reduce the number of terms to L and then allow these parameters to learn freely, making the model capable of replicating any L terms present in the original model by changing λ_i and $\boldsymbol{\zeta}_{i,k}$.

Definition 4.3.3 (Reduced partition model)
For a PQC hypothesis class $\mathcal{F}_{\mathbb{U},M}$*, we define the reduced L-subset partition model as the family of functions* $\overline{\mathcal{F}^L_{\mathbb{U},M}} = \left\{\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}\right\}$ *where each function is given by*

$$\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{i\in[L]} \lambda_i \prod_{k\in[K]} \langle 0| U^{k\dagger}(\boldsymbol{\theta},\boldsymbol{x},\zeta_{i,k}) M_k U^k(\boldsymbol{\theta},\boldsymbol{x},\zeta_{i,k+K}) |0\rangle \tag{4.8}$$

where the unitaries U^k *are those described in Lemma 4.3.1 and we have introduced entirely free parameters* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\zeta}$ *that can be optimised over. This can also be referred to as a "reduced partition model" when L is to be specified later.*

This new model introduces more free parameters, $\boldsymbol{\zeta}$, into our model, fortunately only $2L\times$ *number of cut gates* are introduced.

The reduced partition model can now use the similarity of the terms of the equation 4.8 (they are identical up to the weight vector, $\boldsymbol{\zeta}$) to replicate any subset of terms taken from the partitioned model by simply adjusting the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\zeta}$. This is stated formally in the following theorem.

Theorem 4.3.2
For any PQC hypothesis class $\mathcal{F}_{\mathbb{U},M}$, the L-subset partition hypothesis class $\mathcal{F}^L_{\mathbb{U},M}$ is included in the hypothesis class of the reduced L-subset partition model $\overline{\mathcal{F}^L_{\mathbb{U},M}}$, i.e.,

$$\mathcal{F}^L_{\mathbb{U},M} \subset \overline{\mathcal{F}^L_{\mathbb{U},M}}. \tag{4.9}$$

In other words, if a given classifier can be sufficiently well approximated by considering only L terms, then the hypothesis class of the reduced partition model can do at least as well as this approximation. This is the potential advantage of our model. Additionally, the relaxation from manually picking terms to optimising $\boldsymbol{\zeta}$ allows us to apply gradient based methods, generally yielding much easier optimisation, but in general suffers as the solutions of the relaxation do not encode meaningful solutions of the original problem (which is discrete in nature). However in our case, since we deal with QML, all this achieves is expanding the hypothesis class, where any solution is meaningful, and optimisation (if done completely) can only yield better results with respect to the training error. Although, when expanding the hypothesis class, the problem may become worse generalisation performance, often evidenced by looser/worse generalisation bounds. We analyse these in the next section.

4.4 Generalisation Error

In creating the reduced partition model, we partitioned the circuit and removed terms, which intuitively makes the model simpler. But then, we introduced free parameters, making the model more complicated and increasing the size of the (reduced model) hypothesis class. We will formally study the effect these alterations have in terms of *generalisation error*, defined roughly as the gap between performance on a training set and performance on unseen data from the same distribution. We will only briefly define a few concepts that are needed, readers keen to see a more complete treatment are referred to [73]. We define a supervised learning task on a domain $\mathcal{X}$ and co-domain $\mathcal{Y}$ with a probability distribution over $\mathcal{X}\times\mathcal{Y}$, P, and loss function, $\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$. Supervised learning is the task of outputting a hypothesis, $h \in \mathcal{Y}^{\mathcal{X}}$, such that the risk, $R(h)$ is minimised. We define the risk for a hypothesis h on a continuous space as the expected loss:

$$R(h) = \int_{\mathcal{X}\times\mathcal{Y}} \ell(h(x), y) dP(x, y). \tag{4.10}$$

In practical settings we normally lack access to the underlying probability distribution, so the true risk cannot be evaluated. Instead we are supplied with training data drawn from P, $S = \{(x_i, y_i) \sim P \,|\, i \in [m]\}$, and must settle for evaluating the risk on this finite set. We call this the empirical risk of h with respect to S:

$$\hat{R}_S(h) = \frac{1}{|S|} \sum_{(x_i,y_i)\in S} \ell(h(x_i), y_i). \tag{4.11}$$

Optimising our hypothesis on the training data optimises the empirical risk, which is generally a good proxy for the true risk. The gap between these two risks is bounded by generalisation bounds, specifically by a generalisation gap function g, which can depend on many properties given the specifics of the learning task and hypothesis class. Here we will bound the gap with a function independent of the data distribution, depending only on the the hypothesis class, $\mathcal{F}$, the size of the training set, m, and an acceptable overall failure probability, δ.

More precisely, we will aim for a probabilistic bound on generalisation gap in terms of the risks of the form:

$$P\left(R(h) < \hat{R}_S(h) + g(\mathcal{F}, m, \delta)\right) > 1 - \delta \tag{4.12}$$

Intuitively, the gap has to do with the concept of "overfitting". Simple models tend to have much smaller generalisation gaps. A function which outputs random labels and does no learning has a generalisation gap of 0 but a large empirical risk. In contrast, some complex and large models are found to "overfit" data, where the empirical risk drops to near zero but the generalisation of the model is very poor, with poor performance on data points not seen during training. Since our model contains more parameters than the model we derived it from, we might fear we have slid into the poor generalisation-good empirical risk category.

4.4.1 Encoding Dependent Generalisation Gap

Recall, our objective is to study the generalisation bounds of our model, which attains the form in equation 4.8, where the salient paramters are the number of terms in the summand (L) and the number of blocks in the product (K).

One insightful analysis of generalisation performance is given in [74], we will show that its bounds apply directly to our model. The analysis first imports a result shown in [75, 76], that the function implemented by any

PQC, $f_{\boldsymbol{\theta}}(x)$, is a generalised trigonometric polynomial(GTP):

$$f_\theta(x) = \sum_{\omega \in \Omega} c_\omega(\theta, M) e^{-i\omega x} \tag{4.13}$$

where the effect of all the parameterised gates and the measurement is only reflected in the coefficients $\{c_\omega\}$. The frequencies available in the GTP (Ω) are determined entirely by the input data's encoding strategy, specifically the eigenvalue spectra of Hamiltonians encoding the input data, typically as rotation gates. Further study of the spectra of frequencies, Ω, is available in the aforementioned works.

With a very similar analysis it can also be shown that a GTP of this form exists for each term of our sum: Consider a single term, T_i

$$T_i = \tag{4.14}$$

$$\lambda_i \prod_{k \in [K]} \langle \phi_k | \, U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k+K}) \, | \phi_k \rangle \tag{4.15}$$

this is equivalent to reuniting $|\phi_k\rangle$ and M_k from product form, and combining $\bigotimes_{k\in[K]} U^k(\boldsymbol{\theta}, \boldsymbol{x}, \zeta_{i,k+K}) =: U(\boldsymbol{\theta}, x, \boldsymbol{\zeta}_i)$ into:

$$T_i = \lambda_i \langle \phi | \, U(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_i) M U(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_i') \, | \phi \rangle \tag{4.16}$$

this term is now an inner product of an incredibly similar form to the PQC it is derived from, indeed if the encoding gates are untouched by the partitioning scheme then T_i has the same encoding gates and it can be shown admits a representation as a GTP of the same form, with the exact same spectra, Ω. Our new GTP will contain different (and now possibly complex) $\{c_\omega\}$.

Since each term can be represented as a GTP with the same Ω we are able to combine them into another GTP:

$$\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{j\in[L]} \lambda_j \sum_{\omega\in\Omega} c_\omega(\boldsymbol{\theta}, \zeta_j, M) e^{-i\omega \boldsymbol{x}} =$$

$$\sum_{\omega\in\Omega} e^{-i\omega\boldsymbol{x}} \sum_{j\in[L]} \lambda_j c_\omega(\boldsymbol{\theta}, \boldsymbol{\zeta}_j, M) =$$

$$\sum_{\omega\in\Omega} e^{-i\omega\boldsymbol{x}} c'_\omega(\boldsymbol{\theta}, \boldsymbol{\zeta}, M)$$

with new weights: $\{c'_\omega = \sum_{j\in[L]} \lambda_j c_\omega(\boldsymbol{\theta}, \zeta_j, M)'\}$.

This defines a new GTP of the same degree and the same Ω as the full sized circuit which we originally partitioned. Performing the analysis of type presented in [74] for our circuit gives identical bounds as for the whole (unpartitioned) model.

As our model dramatically differs in the number of terms (which ought to decrease the gap), yet is much more general in the parameters that are free (which should increase the complexity), we see that this bounding technique is quite coarse-grained. In particular, even just pure product models (no entangling gates) which are trivially classically simulatable have the same bounds. The fact that the GTP approach yields somewhat loose bounds was emphasized in [74] and as we have only bounded that bound we must tighten our analysis to achieve a meaningful bound; in the next section we will achieve this.

4.4.2 Term-Based Generalisation Gap

In subsection 4.4.1 we saw that using the analysis technique from [74] the generalisation error analysis for the un-partitioned model matched those of our new model.

The reason for this was that this method inherently only analyzes the way the data is encoded (i.e., how the individual unitaries depend on the input), and this feature is not different between the partitioned and unpartitioned model. In order to obtain bounds which are actually sensitive to the cutting process it is important to examine another approach. We want to analyse an approach that fundamentally considers the increasing number of terms. To this end, in this section we will introduce and bound a complexity measure, the Rademacher complexity, finding that our bound scales linearly in L.

The Rademacher complexity [73] is measure of a function family's ability to assign arbitrary labelling to a set of input data. Given some particular input dataset $S = (x_1, x_2, \ldots, x_m)$ the Rademacher complexity of a function family $\mathcal{F}$ is

$$\mathcal{R}_S(\mathcal{F}) = \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{f\in\mathcal{F}} \sum_{i=1}^{m} \sigma_i f(x_i) \right].$$

The above expectation is over m i.i.d. binary random variables σ with equal chance of being $+1$ or -1. The random variables simulate the random labeling of the data. By maximizing $\sum_{i=1}^{m} \sigma_i f(x_i)$ we identify the classifier, element of the hypothesis class, which intuitively does the best job of matching these random labels on average. Our results hold for any

given dataset so we will omit S and use just $\mathcal{R}(\mathcal{F})$

The main tool we will use is the sub-additive property of the Rademacher complexity [73, 77]:

$$\mathcal{R}(\mathcal{F}+\mathcal{G}) \leq \mathcal{R}(\mathcal{F}) + \mathcal{R}(\mathcal{G}). \tag{4.17}$$

For two families of functions $\mathcal{F}$ and $\mathcal{G}$, where the sum $\mathcal{F}+\mathcal{G} = \{f+g : f \in \mathcal{F}, g \in \mathcal{G}\}$. Taking R_T as the maximum Rademacher complexity of any of the summands in our model. We can bound the Rademacher complexity of our model by $O(R_T L)$, a linear increase with the number of terms in our model. This bound does not directly depend on K (the number of partitioned blocks in our model). However, larger values of K may require larger values of L in order to mimic the behaviour of the unpartitioned model.

Comparing R_T, the Rademacher complexity of a single term of the summand in the model, to the Rademacher complexity of the unpartitioned model presents challenges: In general, a single term of the model is a product of smaller blocks, where the two qubit gates between blocks have been replaced by single qubit gates. Intuition tells us that removing these connecting gates from a circuit should reduce the expressivity, however, cases can be constructed where removing two qubit gates increases generalisation performance (an example is presented in appendix 4.A). Thus it is impossible to simplify the Rademacher bound any further while remaining maximally general i.e. without resorting to circuit specific methods.

4.5 Evaluation and training of reduced partition model

In this section we look at how one can evaluate the circuits, what error this would entail, and how it might be trained, we speculate on a possible feature of partitioned PQCs that might placate the effects of so-called "barren plateaus".

4.5.1 Evaluation

Evaluation of the reduced partition model is a non-trivial task, the terms are composed not of expectation values (which can be evaluated with simple circuits) but of inner products, with different unitaries on either side of the observable. Fortunately this is not an insurmountable problem. To evaluate these inner products we can employ the Hadamard test shown in

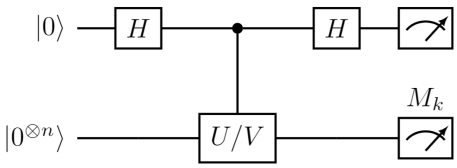


Figure 4.2: The hadamard test to be used for calculating the real component of some $\langle 0| UM_kV |0\rangle$. The controlled U/V circuit implements U if the control is 0 and V if the control is 1. It is important to note that the controlled U/V circuit only requires on control on a few gates, since only the partition gates differentiate U and V most gates are identical and do not require control. This circuit can be modified to calculate the imaginary component as described in [12]

figure 4.2. The most challenging component of this circuit is the application of a controlled-U/V, naively this would require controlled gates for every gate in U and in V. Fortunately this is not the case. Since U and V differ only by the partition gates, the controlled-circuits can be constructed with controlled operations only on these partition gates, which is a small subset of the total number of gates in the circuit.

As with all NISQ applications we must inspect how our algorithm will perform on a noisy device. By bounding the variance we find the noise scaling very reasonable.

First, let us replace the non-random inner products,

$$\langle \phi_k | U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k+K}) | \phi_k \rangle ,$$

with random variables $X_{i,k}$ which are unbiased estimators of the inner product (that is $\langle \phi_k | U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta}, \boldsymbol{x}, \zeta_{i,k+K}) | \phi_k \rangle = \overline{X_{i,k}}$ where the bar now represents the expectation value). These random variables represent an estimation of the inner product with s shots on a quantum computer. We are interested in bounding the probability that the difference between the estimate and the average exceeds some ϵ by δ.

$$P\left(\left| \sum_{i\in[L]} \lambda_i \prod_{k\in[K]} X_{i,k} - \sum_{i\in[L]} \lambda_i \prod_{k\in[K]} \overline{X_{i,k}} \right| > \epsilon \right) \leq \delta \tag{4.18}$$

We can achieve this bound by considering the variance. We will assume the observable and each λ are bounded by 1, although we will comment on how this is easily generalised. We find the variance scales with the number

of shots:

$$\sigma^2 \leq \frac{4K^2L}{s}$$

by [78]. Equation 4.18 is satisfied when we have $s = \frac{4LK^2}{\epsilon^2\delta}$ shots by Chebyshev's inequality. To generalise this to an observable or variance greater than one, note that the argument of the probability in equation 4.18 can simply be re-scaled as both of these elements are linear, this in-turn re-scales the variance providing a bound.

4.5.2 Training

Training with a gradient based approach is easy to apply in our model too. The derivative distributes on terms of the sum and can be evaluated by applying the chain rule to the product in each term. Indeed since most parameters appear in only one gate on one qubit on one side of the partition, the chain rule evaluates to 0 on all but 1 element of the product. Evaluating the gradient then takes at most L times the number of evaluations required to evaluate the gradient of one of the smaller circuits. In this case we find that evaluating the gradient for any parameter, $\boldsymbol{\theta}^t$, that exists only in the k'th partition is:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\theta}^t} \bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x}) = \\
&\sum_{i\in[L]} \lambda_i \frac{\partial}{\partial \boldsymbol{\theta}^t} \prod_{k\in[K]} \langle\phi_k| U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k}) M_k \\
&\qquad\qquad U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k+K}) |\phi_k\rangle = \\
&\sum_{i\in[L]} \lambda_i \frac{\partial}{\partial \boldsymbol{\theta}^t} \langle\phi_{k'}| U^{k'}(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k'}) M_{k'} \\
&\qquad\qquad U^{k'}(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k'+K}) |\phi_{k'}\rangle \times \\
&\prod_{k\in[K]\setminus k'} \langle\phi_k| U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k+K}) |\phi_k\rangle
\end{aligned} \tag{4.19}$$

The same applies for the ζ parameters. In many instances the gradient can be made easier to compute, since we have often already evaluated the non-derivative expression before looking for the gradient most of the circuit evaluations are already done, with only the derivative expression for a single inner product requiring a new evaluation. Which can be done in the standard manner (e.g. parameter shift rule [79]).

4.5.3 Barren Plateaus

A well studied problem [80] with PQCs is the "barren plateaus" phenomenon, where large parts of the parameter landscape have an exponentially small gradient, effectively crippling optimisation. This is a manageable problem for currently implementable PQCs due to their limited size, but as PQCs become larger (and their gradient decreases) the problem intensifies [80]. While our model is not immune to barren plateaus we may be able to reduce their effect on our model relative to the size of their effect on the unpartitioned circuit.

Each term of our model is a multiplicative separable function (it can be written as: $f_T(x_1, \ldots, x_K) = f_1(x_1) \times \ldots \times f_K(x_K)$, where f_i is an inner product and, x_i is the input to the inner product, including the data and weights) we simplify to assuming x is a single parameter, for illustrative purposes. To calculate the gradient we apply the chain rule to the product, for most architectures any particular parameter will only appear on one block of the partition, then one term of the chain rule will be non zero $\partial_{x_i} f_T(x_1, \ldots, x_K) = f_1(x_1) \ldots (\partial_{x_i} f_i(x_i)) \ldots f_K(x_K)$.

The gradient is thus determined by multiplying together the many amplitudes stemming from the subcircuits of the sum with this lone gradient term (equation 4.19). Two aspects may make this overall gradient small: First the gradient may be small as it is a PQC and is prone to barren plateaus, however the individual subcircuits generically have larger gradient than the full unpartitioned circuit as they are smaller [80] (i.e., the barrenness of the plateaus heavily depends on the number of qubits in the circuit). Second, the multiplication with other terms may cause it to decay to zero as we are dealing with a product of terms which are absolute value below 1, the product then decays exponentially in the number of multiplicative terms to some small number. However in our case we are not directly facing this radically smaller number, we fundamentally have more information about the gradient, knowing the total gradient, but also the terms that are combined to form it. We know the effect that varying any of these subterms has on the gradient of the complete circuit. One possible use of this information is to identify which term is driving the gradient to a small value, and to revert its parameters back to an earlier instance which we have stored in memory, through this method the impact of barren plateaus could be mitigated. A technique similar to [81] could be developed, to avoid low gradient directions, but utilising the more information present in our case.

There is quite a bit of research on *additive* separable functions, which may transfer to our case [82]. This could lead to significantly easier training.

Experiment	Model Type	Accuracy(A)/ MSE
MNIST handwriting	Neural Net	100%A
	RPM	96.4%A
Approximating larger PQC	Neural Net	0.424MSE
	RPM	0.0322MSE

Table 4.1: A summary of the main numerical findings in this section, we report our model's (RPM) performance on handwriting recognition and on simulating the output of a larger PQC. We also train a neural network as a comparison.

We plan to develop this method of training in a follow-up work.

4.6 Numerics

In the previous sections we laid out a model with considered theoretical underpinning, in this section we will demonstrate that model's basic utility by showing it can learn a simple large problem, the MNIST handwritten digit recognition, by utilising an ansatz much larger than the computer it has simulated access to. We also present an experiment designed to test if an adequate approximation of a random circuit output can be made with much fewer terms, we then apply our model on the same random circuits output to test its performance on synthetic data.

4.6.1 A Large Problem: Handwriting

Reading handwritten numbers is one of the most basic tasks in undergraduate machine learning courses. The MNIST [83] data set presents a relatively simple task, identify which digit is written in an 28×28 pixel image, but even this simple task is difficult for current generation quantum machines due to its high dimensionality, with quantum attempts only succeeding recently through careful encoding of the problem (e.g. in [65]). Often dimensionality reduction techniques such as principal component analysis are applied [84] but for a simple problem like MNIST handwriting this reduces the learning problem to a triviality. Here we preserve the learning problem by downsampling the image to just 64 pixels, which is importantly still human readable. Here we will show that even simple cases of our model perform adequately and by increasing L (the number of terms of our model) we increase that performance.

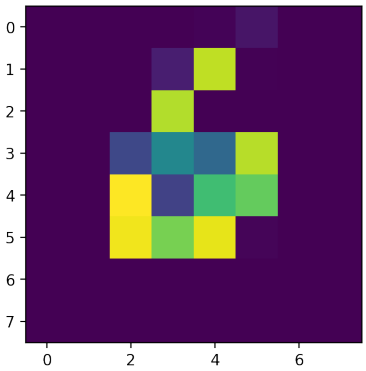


Figure 4.3: An example datum of the handwriting task, the number 6. The picture is then cut into 8 elements (as given by its columns) with each element as input to a different PQC. This resolution data was chosen so as to be fine enough to be human readable.

For purposes of comparison we reduce the problem to differentiating 3 and 6, as in [85]. Our model is based on an 8 block partitioning of the 64 qubit, depth 3 hardware efficient ansatz (of the same form as in [72]) into 8 qubit blocks, a model which would normally be far outside of our computational power. An unseen validation set is evaluated at every step of training and the results are shown in figure 4.4. The final training loss (MSE), testing loss (MSE) are shown in the table 4.2, we also apply a step function to the output (to convert its real valued output into a binary label) and list its accuracy.

Data augmentation (skews and rotations) were used to generate more data for the model. Without this augmentation high L terms began to overfit, increasing the training performance while decreasing the validation performance. With data augmentation we can see that our model is behaving well, even in the 1 term case we find that it selects a good arrangement of weights, although with relatively few additional terms the performance increases, for contrast to run this model using the complete circuit partitioning scheme would require the evaluation of over 46000 subcircuits. A neural network with a convolution layer and a single dense 128 neuron hidden layer is provided for comparison. We must consider that our results are on MNIST handwriting, which is known to have many problems and cannot be used to claim that our model excels on all similarly large tasks [86].

L	$\text{loss}_{training}$	$\text{loss}_{testing}$	acc%
1	0.0521	0.0499	92.8
3	0.0488	0.0465	92.8
5	0.0479	0.0454	94.7
10	0.0432	0.0422	96.2
20	0.0359	0.0341	96.4
Neural Net	0.0025	0.0031	100

Table 4.2: Final loss on training/validation sets and ultimate accuracy of the reduced partition model on the handwriting recognition task. Different values of the hyperparameter L are listed. A neural network is also trained on the task and found to perform very well.

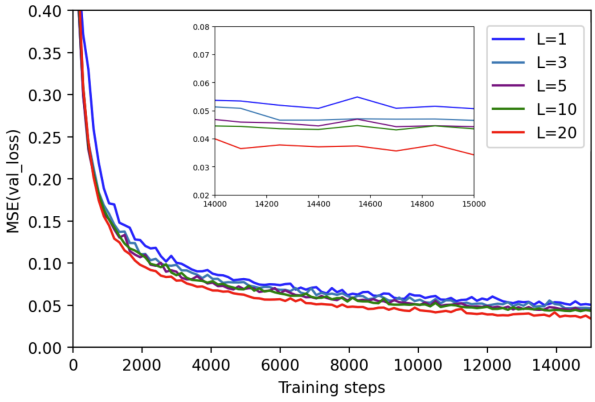


Figure 4.4: The loss on an unseen data set evaluated alongside the training of a reduced partition model to recognise handwritten digits. Increasing the number of terms has a positive effect on the ability of the model.

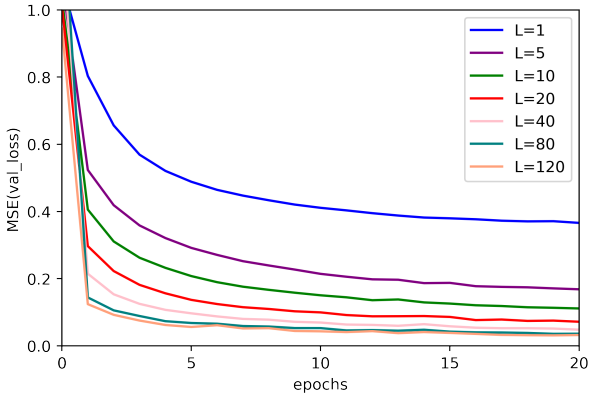


Figure 4.5: The performance of our model on replicating the output of a un-partitioned VQC when θ is fixed to that of the un-partitioned model, and only the parameters meant to replicate the partitioning process are trained. The amount of terms included is varied according to the Legend in the top right.

4.6.2 Tests on Synthetic Data

This work is built around the assumption that many terms in the partitioned equation for a given circuit are redundant, and a good approximation of the complete circuit can be made by our model. In this section we test this assumption with our first experiment, and then test our complete model on learning a synthetic data set in the second and third experiments.

We take a width 10, depth 3 instance of the hardware efficient ansatz with random weights. Using this circuit we generate a synthetic data set by recording its output on 10000 random inputs, we normalise these outputs to to a mean squared average of 1. We then instantiate a modified version of our model corresponding to a partitioning of the full circuit into 2 blocks of width 5. The model is modified from the general model we have described above by fixing $\boldsymbol{\theta}$ (the weights present from the unpartitioned PQC) and only training $\boldsymbol{\zeta}$ and $\boldsymbol{\lambda}$ (the weights we introduced when creating the model). This modification allows us to examine directly our claim that introduction of the free parameters, $\boldsymbol{\zeta}$ and $\boldsymbol{\lambda}$, is sufficient to approximate the output of the full PQC without evaluating the many subcircuits that would be required in theorem 4.3.1. After this experiment we free θ (apply the full model) and examine the increased performance this gives us.

The results of our experiment are shown in figure 4.5. The benefits of increasing L are more apparent than in the digit recognition experiment,

L	Final validation MSE
1	0.366
5	0.168
10	0.111
20	0.0717
40	0.0481
80	0.0362
120	0.0322
Neural Net	0.424

Table 4.3: Results of experiment comparing the validity of our assumptions. We fix the value's of θ and set the reduced partition model to learn the output of a larger PQC, to simulate the larger PQC exactly using [12] would require $L = 16,777,216$, the model is only able to select which parameters to put on the gates resulting from the partition. Different values of the hyperparameter L are listed for comparison. A neural network is also trained and performs poorly. Results are averaged over 5 runs.

we can see better approximations being made at higher L. For some applications more accuracy might be required, it seems increasing L further will continue improve this accuracy. Noteably all L considered are orders of magnitude below the amount of terms or circuits needed to apply the existing partitioning schemes. The final mean squared error for unseen data averaged over 5 random data sets is presented in table 4.3.

Where we have included a neural network with a single dense hidden layer of 256 neurons for comparison purposes, other neural network architectures (1 and 2 hidden layers were tried, with 64 and 256 neurons per layer for each) were tried without meaningful improvement, although it is possible that with thorough tuning these architectures or others could be made to perform strongly.

The previous results are sufficient to show that the training of just the parameters $\boldsymbol{\zeta}$ and $\boldsymbol{\lambda}$ can lead to models with substantially fewer terms, L, while still sufficiently approximating the full circuit in this instance. This approximation was achieved with just the training of $\boldsymbol{\zeta}$ and $\boldsymbol{\lambda}$, while fixing the $\boldsymbol{\theta}$ to those that were used to generate the data. However it is not clear, a-priori, that the reduced model should use the same $\boldsymbol{\theta}$ parameters to best mimic the full model. We now allow $\boldsymbol{\theta}$ to deviate from that of the generating PQC, the resulting mean squared error for unseen data after 20 epochs is presented in table 4.4.

L	Final validation MSE
1	0.176
5	0.112
10	0.0855
20	0.0600
40	0.0434
80	0.0334
120	0.0289
Neural Net	0.424

Table 4.4: Mean squared error of the reduced partition model on learning the output of a larger PQC, unlike table 4.2 all parameters are now free and we can directly test the RPM's capabilities on this task. Different values of the hyperparameter L are listed for comparison. A neural network is also trained and performs very poorly. Results averaged over 5 runs.

This improvement in performance is unsurprising as the unrestricted model includes the hypothesis of the model without training $\boldsymbol{\theta}$, however it was not clear before the experiment that the model would be able to find this higher performance, as the introduction of more parameters may have created too many local optima for efficient optimisation. On the other hand we may have expected a larger increase in performance, as $\boldsymbol{\theta}$ makes up the majority of parameters, we should expect releasing $\boldsymbol{\theta}$ to correspond to a big increase in performance. The lack of this increase could be taken as weak evidence that our approximation (that a smaller set L can approximate the output of the whole circuit) to be relatively accurate in this case, even without retraining $\boldsymbol{\theta}$.

Finally we use the synthetic data set as a training set for our model, with random initialisation of weights. This third experiment allows us to test our models performance on a task which a classical algorithm (the neural network) performs poorly on, without prior knowledge of good parameters.

L	Final validation MSE
1	0.183
5	0.113
10	0.0944
20	0.0703
40	0.0540
80	0.0357
120	0.0362
Neural Net	0.424

These performances are strong and comparable to the previous two experiments, where $\boldsymbol{\theta}$ was given, showing that our model performs well on this task, much better than the neural network we compare it to. This final experiment is an excellent demonstration of our model as it would be deployed, and demonstrates that it can learn a non-trivial task where a higher number of qubits would naively be required.

4.7 Conclusion and future work

In this work we applied previously developed circuit cutting techniques to parameterised quantum circuits. While it is obvious that this naive approach used too many circuit evaluations to be computationally practical we noted there may exist a smaller set of circuits which would sufficiently approximate the original circuit, although we speculate that finding it would itself be computationally intractable even if it did exist. Instead we proposed a new model based on the relaxation of fixed gates into parameterised gates, such that all circuits were identical up to the weights of these newly parameterised gates. We showed our models hypothesis class contained the relevant unparameterised hypothesis class, that its generalisation error was well behaved and then went on to test it experimentally. The first experiment showed the model was capable of tackling large problem sizes (handwriting). We also tested the ability of a parameterised subset of circuits of the partition to approximate the full unpartitioned output of a random circuit and found a very satisfying approximation, although a larger amount of terms was needed than with the handwriting task, suggesting a link between the problem and the number of terms needed to achieve a given accuracy.

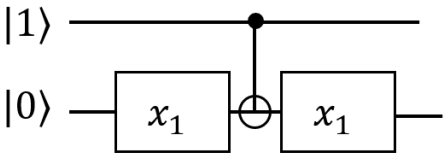
Further work is needed in establishing how many terms (L) might be required for any given task, and what factors influence this requirement. Future work could also focus around the application of this model, testing

it out on larger cutting edge problems, or on achieving higher accuracy. Improvements to the model could come from a development of a robust training procedure to avoid barren plateaus (Section 4.5) or from integrating our work with some of the excellent work already done on improving divide and conquer schemes (Section 4.2). Our work has opened the door for experimentation with much larger "partially quantum" models both implicitly as we have done here, but potentially explicitly, integrating more classical resources into a quantum machine learning setting.

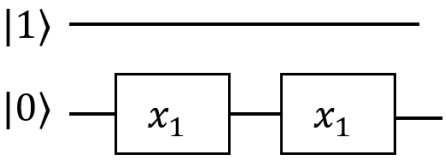
4.A Example of increased complexity after the removal of a 2-qubit gate

In this appendix, we will see that in some highly manufactured cases removing a two qubit gate can lower expressivity.

Consider the following circuit:



Where the encoding gates are Ry. With the CNOT in place the state vector at the end will always be $|11\rangle$, regardless of the input vector. Removing the CNOT entirely produces the following circuit:



With the CNOT removed the final state is $\cos(x_1)\,|10\rangle + \sin(x_1)\,|11\rangle$, which depends on the input, x_1. In this way it is clear how a circuit making use of this (rather pointless) gate would increase in expressivity when a CNOT is removed. No one would realistically propose this circuit, but its existence prevents blanket statements about the effect of removing two qubit gates on complexity and therefore generalisation.

4.B Proofs of theorems

Here we provide proofs of theorems too long to provide in the main text of this chapter.

Theorem (Partitioned model)
For every function $f_{\boldsymbol{\theta}} \in \mathcal{F}_{\mathbb{U},M}$ and qubit partition $\{B_k\}_{k\in[K]}$ with observable $M = \bigotimes_{k\in[K]} M_k$, there exists a set of coefficients $\{c_i\}_{i\in[T]}$ and unitaries $\tilde{\mathbb{U}} = \{U^{i,k}(\boldsymbol{\theta},\boldsymbol{x}), U'^{i,k}(\boldsymbol{\theta},\boldsymbol{x})\}_{i\in[T],\ k\in[K]}$ (where each $U^{i,k}$ and $U'^{i,k}$ acts on n_k qubits) which can be combined in a function:

$$\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{T} c_i \prod_{k=1}^{K} \langle 0| U'^{i,k\dagger}(\boldsymbol{\theta},\boldsymbol{x}) M_k U^{i,k}(\boldsymbol{\theta},\boldsymbol{x}) |0\rangle , \tag{4.20}$$

such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \tilde{f}_{\boldsymbol{\theta},M}(\boldsymbol{x})$ for every $\boldsymbol{\theta},\boldsymbol{x}$. For arbitrary gates the number of terms T grows as 16^r, where r is the number of gates across the partition, but for cut gates with known Schmidt number T is the product of the Schmidt number squared of each cut gate.

Proof. For any given circuit, U, [12] provides a decomposition of the form:

$$U = \sum_{i=1}^{T} a_i \prod_{k=1}^{K} U^{i,k}$$

by writing two qubit gates in the schmidt decomposition. Two qubit gates have schmidt rank of between 2 and 4 [70], thus any two qubit can be expressed as a sum of 4 tensor product single qubit gates. To express an expectation value in this form requires decomposition on both U and $U^\dagger$, using the linearity of the the expectation value we arrive at 16^r inner products for r two qubit gates. Applying this scheme to every element of $\tilde{\mathbb{U}}$ gives us the final statement. □

Lemma
For every $\tilde{f}_{\boldsymbol{\theta}} \in \mathcal{F}^L_{\mathbb{U},M}$, there exists a set of unitaries $\{U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta})\}_{k\in[K]}$ and parameters $\boldsymbol{\lambda},\boldsymbol{\zeta}$ defining a function:

$$\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{i\in[L]} \lambda_i \prod_{k\in[K]} \langle 0| U^{k\dagger}(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k}) M_k U^k(\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{\zeta}_{i,k+K}) |0\rangle ,$$

such that $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(\boldsymbol{x})$ *for every* $\boldsymbol{\theta}, \boldsymbol{x}$ *and for every observable that can be written as tensor product on the elements of the partition* $M = \bigotimes_k M_k$.

Proof. This is a simple extension of the last theorem. For fixed i, k we have to find $U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k})$ such that

$$U^k(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\zeta}_{i,k}) = U^{i,k}(\boldsymbol{\theta}, \boldsymbol{x})$$

for some $\zeta_{i,k}$. We know that the cutting scheme in [12] replaces the site of removed two qubit gates with single qubit unitaries, parameterising the difference between the unitaries proves the result.

□

Theorem

For any PQC hypothesis class $\mathcal{F}_{\mathbb{U},M}$*, the L-subset partition hypothesis class* $\mathcal{F}^L_{\mathbb{U},M}$ *is included in the hypothesis class of the reduced L-subset partition model* $\overline{\mathcal{F}^L_{\mathbb{U},M}}$*, i.e.,*

$$\mathcal{F}^L_{\mathbb{U},M} \subset \overline{\mathcal{F}^L_{\mathbb{U},M}} \tag{4.21}$$

Proof. By lemma 4.3.1 we know that for all $\tilde{f}_{\boldsymbol{\theta}} \in \mathcal{F}^L_{\mathbb{U},M}$ there exists $\boldsymbol{\zeta}$ selecting a function $\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}} \in \overline{\mathcal{F}^L_{\mathbb{U},M}}$ such that $\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(x) = \tilde{f}_{\boldsymbol{\theta}}(x) \forall x$.

□

CHAPTER 5

All this for one Qubit?

5.1 Introduction

The near-term deployment of advantageous quantum algorithms is currently constrained by available hardware. Among other limitations, modern machines simply do not have enough qubits for the most interesting algorithms. In an attempt to augment modern machines, several cutting schemes [3–7, 7–11] have been proposed that partition a given quantum circuit into smaller blocks. Each block can be run independently and then combined to simulate the output of the original circuit. We refer to these techniques collectively as “Circuit Cutting" (CC). One application of these circuit-cutting schemes was given in the previous chapter.

While the potential value of these schemes is clear, their practical value is limited by their computational cost: the number of circuit evaluations required grows exponentially with the number of two-qubit gates between partitioned blocks. For many algorithms, each qubit requires a polynomial number of two-qubit gates connecting it to the rest of the circuit, preventing useful application of CC to these cases. Applications of CC are instead relegated to a secondary role, such as augmenting connectivity by adding virtual connections [5]. Broader application of CC can therefore only be achieved if the number of terms can be reduced, one such route is by a computational cost of a scheme that scales in e.g. number of qubits

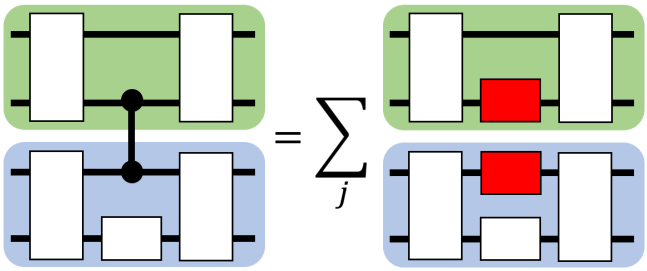


Figure 5.1: Circuit cutting schemes can be used to simulate computations with more qubits than a user has access to. Here a circuit cutting scheme rewrites two-qubit gates as sums of single-qubit gates, allowing the 4-qubit circuit to be expressed as a sum of tensor product two-qubit circuits, those two-qubit circuits can then be evaluated on a smaller machine. This chapter focuses only on "cut-local" schemes, i.e. ones that only modify the circuit at the sites of partition-crossing two-qubit gates, replacing the gates with a sum of single-qubit unitaries.

removed, instead of two qubit gates cut.

While it is clear that some partitions will require a super-polynomial number of circuit evaluations if quantum computers provide a computational advantage, **BPP≠BQP** (a simple example of this is given in the footnote [1]), this super-polynomial requirement could still produce achievable runtimes if it scaled in a different parameter, such as the number of qubits in the blocks of the partition. In some ways, the size of the largest block is a more natural scaling parameter, for instance, the limited Hilbert space dimension limits the maximum amount of entanglement between blocks (a key ingredient in quantum advantage [87, 88]).

The size of the Hilbert space is a basis for the proofs of circuit cutting scaling requirements in [7, 9], where it is shown that an exponential number of terms are needed to simulate large and highly entangled states.

While this argument bounds circuit-cutting techniques which scale with number of two-qubit gates, it does not, for example, bound CC schemes which take into account fixed inputs or scale in other parameters [8].

[1] Cutting the circuit in half, then repeatedly cutting the subcircuits generated by this cut in half would require only $\lceil(\log_2(n))\rceil$ rounds to reduce an n-qubit circuit to a combination of 1-qubit circuits. If each round produces at most A subcircuits only $A^{\log(n)}$ circuit evaluations are needed to reduce the n-qubit circuit can be reduced to a polynomial-sized set of classically simulatable 1-qubit circuits. If the cutting procedure produced less than some pseudo-polynomial number of terms each time, A, then a classic simulator exists using pseudo-polynomial time.

This chapter investigates if there could exist such a circuit cutting scheme: one whose computational overhead would scale polynomially with the number of inter-partition gates, at the cost of an exponential scaling in the size of the smallest partitioned block. We show that when the circuit cutting scheme is limited to local modifications (i.e. it can only modify partition-crossing two-qubit gates; removing this assumption would make any formal statements dramatically more difficult to prove [2]) any circuit cutting scheme that can efficiently remove a single qubit (create a $(1, (n-1))$ partition) would imply **BPP**=**BQP**. These results demonstrate that circuit-cutting schemes can not be broadly efficient, regardless of what tricks are applied, and that they do not represent a shortcut to quantum advantage. In relation to this thesis, this chapter demonstrates that the heuristic circuit-cutting-esque machine learning algorithm given in the previous chapter cannot work on all circuits (when phrased as machine learning problems).

5.2 Background

Circuit cutting (CC) schemes [3–7, 7–11] (sometimes referred to as circuit partitioning or circuit knitting) are a class of methods designed to reduce the demands on a quantum computer when trying to implement a large quantum circuit. Existing schemes either make multiple calls to the device [3–6] or link multiple devices with classical communication [7, 9] to simulate the larger circuit.

Each of these circuit-cutting schemes works slightly differently, for simplicity, we will focus on a generalization of the formalism presented in [3]. This generalization involves decomposing a unitary into a number of smaller unitaries. Here, decomposing means expressing a given n-qubit unitary as a sum of tensor products of two fewer-qubit unitaries:

$$U = \sum_{i}^{L} \alpha_i U_i' \otimes U_i'', \tag{5.1}$$

We discuss how our results generalize to alternative schemes (e.g. ones that decompose tensor-networks [4] or superoperators [5]) later in this chapter.

[2] As we discuss later, local schemes effectively mean we do not allow significant semantic-preserving circuit rewritings; allowing circuit rewritings is more powerful, but also leads to the (NP-hard) problems of finding minimal or otherwise simplest circuits, making proofs exceptionally difficult. Hence we focus on the simpler case here.

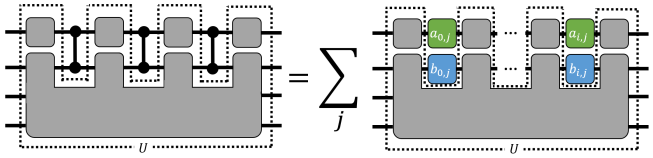


Figure 5.2: Figure depicts a cut-local circuit cutting scheme being applied to our quantum comb formalism. The comb is designed to specify the whole circuit except for some "gaps", which are then filled by input gates, in this case, Controlled-Z gates, that can add connectivity between the first qubit and the other $(n-1)$-qubits. When a cut-local circuit cutting scheme is applied to the circuit, the connecting two-qubit gates in the gaps become single-qubit operators. The resulting unitary is now a tensor-product allowing the first qubit and the other $(n-1)$-qubits to be run separately, reducing the required width of quantum computer. To go beyond cut-local schemes, one would allow U to depend on j.

All these existing approaches to circuit cutting have a drawback: they incur a super-polynomial scaling in the number of connections, k, crossing the partition, i.e. $L \geq c^k$ for some $c \geq 2$. This chapter addresses an intriguing question: could there exist a scheme capable of partitioning an n-qubit circuit into an m-qubit block and an $(n-m)$-qubit block with super-polynomial scaling only in m, while remaining polynomial in k and n, in other words, we seek a scaling $L \in O(c'^m \times \text{poly}(n,k))$. We find that even considering the simplest case, where $m = 1$, is sufficient to show it is not possible.

Conflicting intuitions surround the possibility of a $\text{poly}(n,k)$ scheme for the $m = 1$ case. On one hand, the addition of a qubit doubles the dimension of the relevant Hilbert space, making it unclear how to simulate the larger Hilbert space with access to only the smaller one. On the other hand, existing limitations of circuit cutting rely on simulating states with substantial entanglement between the blocks to show that $L \geq 2^k$. However, in the $m = 1$ case, this entanglement is heavily limited, breaking the assumptions behind these limitations. Notably, if we simplify the goal to expressing equation 5.1 as a sum of *arbitrary linear operators*, it clearly only requires 4 terms to satisfy equation 5.1 [3]. This chapter resolves these uncertainties by showing that when the circuit cutting scheme is only allowed to make local modifications, an exponential number of terms in k is necessary, regardless of how many qubits are removed.

[3] This is the upper bound on the Schmidt rank when partitioning a 2-dimensional subspace.

This brings us to a key property shared by all existing circuit cutting schemes, which we refer to as "cut-locality". The idea behind cut-locality is that when modifying a gate the resulting subcircuit only differs at the site of the removed gate. In [3] this is a two-qubit gate replaced locally as the sum of one-qubit gates (see Figure 5.1).

To formally treat cut-local schemes it is useful to use a variation of the quantum comb formalism [89]. We are only interested in using the formalism to partition a single qubit from our circuit, thus we will slightly modify the definition of a quantum comb. Informally our quantum comb is a unitary with G "gaps" where gates can be plugged in, the unitary does not have connections between the first qubit and the rest of the qubits but by putting in two-qubit gates to these gaps we can create a connected unitary. An example is shown in Figure 5.2. We define the quantum comb as a map, $U(\cdot)$, taking in two-qubit unitaries, G_i, and returning a fixed n-qubit unitary with G_i in the gaps as described.

Our ultimate goal is to bound cut-local schemes, which would transform a quantum comb with entangling two-qubit gate arguments into the sum of quantum combs with tensor product arguments:

$$U(\ldots, G_j, \ldots) = \sum_{i=0}^{L} \alpha_i U(\ldots, a_{i,j} \otimes b_{i,j}, \ldots) \tag{5.2}$$

where $a_{i,j}, b_{i,j} \in SU(2)$ (single qubit unitaries), $\alpha_i \in \mathbb{C}$ (some coefficient) and $G_j \in SU(4)$ (two-qubit gates). If a given quantum comb with given two-qubit unitary inputs can be represented as the sum of quantum combs with tensor product inputs in at most L terms, as in equation 5.2, we say there exists an *L-term partitioned quantum comb* representation.

We have yet to define whether the quantum comb is equal to a specific computation, with known input and observable (i.e. classically specified and fixed), or to the unitary itself (so the decomposition does not benefit from considering specific input states or measurements, and must correctly apply to all inputs and measurements). The following paragraphs will address both of these cases; we show the unitary case is simple via linear algebra, the fixed-input-output case is more challenging, requiring a complexity-theoretic argument.

5.3 Bounds on the optimal scheme

Our results are structured into two groups: The first group, Lemma 5.3.1 and Theorem 5.3.1, shows that when the input and observable are fixed

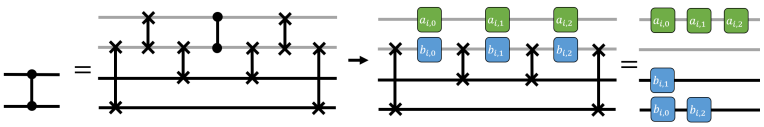


Figure 5.3: Demonstration of a gadget used in proofs of theorems 5.3.1 and 5.3.2. The gadget adds two ancilla qubits (shown in grey) to move any non-SWAP two-qubit gate to these two ancilla qubits. Applying any local circuit cutting scheme to separately partition the first qubit then leaves a circuit of only SWAP and single-qubit gates. This can then be classically rewritten in polynomial time as a sum of circuits of single-qubit unitaries. To evaluate this circuit classically requires only polynomial time, this is used to reach a contradiction to prove theorem 5.3.1. Polynomial sums of single-qubit unitaries can only have polynomial Schmidt rank, this is used to reach a contradiction to prove theorem 5.3.2.

it is technically possible to partition a single qubit from the rest of the circuit in polynomially many terms, but that if such a decomposition could be found in polynomial time then **BPP**=**BQP**. Our second group, centered on Theorem 5.3.2 shows that even with unlimited runtime a local circuit cutting scheme cannot find a decomposition of a given circuit in polynomially many terms if we force the same decomposition to apply for every input and observable.

Our first result, showing that a decomposition can be found, is only true in a technical sense and not informative to practical use.

Lemma 5.3.1
Given a circuit expressed as a quantum comb with fixed gates, $U(G_1, \ldots)$, a known (i.e. there is a known succinct classical description) input, $|\phi\rangle$ and known observable, M, the associated expectation value can be expressed with a 1-term partitioned quantum comb in the same expectation value:

$$\langle\phi| U^\dagger(G_1, \ldots) M U(G_1, \ldots) |\phi\rangle = \tag{5.3}$$

$$\alpha_0 \langle\phi| U^\dagger(a_{0,0} \otimes b_{0,0}) M U(a_{0,0} \otimes b_{0,0}) |\phi\rangle\,, \tag{5.4}$$

where $a_{0,0}, b_{0,0} \in SU(2)$ (single qubit unitaries), $\alpha_0 \in \mathbb{C}$ (coefficient).

Proof. The outcome of equation 5.3 is equal to some real number, γ. If $\gamma = 0$, set α_0 to 0 and $a_{0,0}$ and $b_{0,0}$ to any single qubit gates. The equality holds proving this case.

If $\gamma \neq 0$, by [3] it is known that there must exist some input $a_{0,0}$ and $b_{0,0}$ that produces a non-zero output of equation 5.4. This non-zero output

can then be rescaled with some $\alpha_0 \in \mathbb{C}$ to achieve the equality. □

It is easy to see that if a scheme existed to produce this 1-term partitioned quantum comb, then we could iteratively apply it to the whole circuit and obtain a classical algorithm with polynomial runtime to compute any quantum circuit, implying **BQP** = **BPP**. The following theorem then demonstrates that *finding* this partition must in some way contain the hardness of **BQP**, indeed it shows that the existence of any polynomial-time circuit cutting algorithm capable of finding this polynomial termed partitioned quantum comb would imply **BQP** = **BPP**.

Theorem 5.3.1
If there exists a polynomial time classical algorithm that takes an arbitrary input circuit expressed as a quantum comb with fixed gates, $U(G_1, \ldots)$, a known input $|\phi\rangle$ and a known observable M, and returns the arguments of an L-term partitioned quantum comb, $a_{i,j}, b_{i,j} \in SU(2), \alpha_i \in \mathbb{C}$ for $L \in poly(k, n)$, such that:

$$\langle\phi| U^\dagger(G_1, \ldots) M U(G_1, \ldots) |\phi\rangle =$$
$$\langle\phi| \sum_i^L \bar{\alpha_i} U^\dagger(a_{i,0} \otimes b_{i,0}, \ldots) M \sum_i^L \alpha_i U(a_{i,0} \otimes b_{i,0}, \ldots) |\phi\rangle$$

then **BQP** = **BPP**.

Proof. Given an algorithm that can partition a single qubit as described (call this algorithm $\mathcal{A}$) we provide an efficient classical simulator.

Given an n-qubit input circuit V written in some standard gate set (e.g. H, CNOT, T), replace every existing two-qubit gate anywhere in V with the gadget shown in Figure 5.3, creating a circuit of only swap gates and single-qubit gates everywhere except for arbitrary 2 qubit gates between the first 2 qubits.

Applying $\mathcal{A}$ to separate the top qubit produces a circuit of only SWAP and single qubit gates, which is classically simulatable in poly(n) time [1]. □

This is our main result: put simply, local schemes cannot partition even a single qubit without paying an exponential cost somewhere. Extending this result to the impossibility of separating l qubits is just a matter of "padding" the gadget with $l - 1$ qubits which do not interact with the rest of the circuit. Note that this result transfers to the task of approximating (rather than exactly recreating) the output with an L-term partitioned quantum comb as **BQP** is robust to approximations of outputs. We will

also describe how this result can be extended to other local circuit cutting schemes in following paragraphs.

While the condition **BQP** $\neq$ **BPP** is a reasonable requirement, it is not clear if it is necessary. We show that by forcing one decomposition to apply for all inputs and measurements (which is equivalent to demanding the unitaries are the same, and thus could be derived from existing work such as [7]) we can show unconditionally that it is impossible to partition one qubit from an n qubit circuit with only polynomially many terms.

Theorem 5.3.2
There exist quantum circuits expressible as a quantum comb with input gates, $U(G_1, \ldots)$, such that for all $L \in O(poly(n))$ and inputs $\alpha_i \in \mathbb{C}, a_{i,j}, b_{i,j} \in SU(2)$,

$$U(G_1, \ldots) \neq \sum_{i}^{L} \alpha_i U(a_{i,0} \otimes b_{i,0}, \ldots).$$

Proof. Define C as a circuit that generates $n/2$ Bell pairs from the $|0\rangle^{\otimes n}$ state (if n is odd, generate $n-1$ bell pairs). We can apply the rewriting gadget in Figure 5.3 to C, call this new circuit C'.

Assume towards contradiction that there exists a polynomial-L term quantum comb implementing the same unitary as C', by separating the first qubit we have created a sum of L tensor product circuits. As shown in Figure 5.3, each of these circuits acts locally on every qubit, thus each individual circuit has an operator Schmidt rank of 1 across any partition. Summing over L tensor product terms produces an operator of Schmidt rank at most L. Applying this circuit to the Schmidt rank 1 input state, $|0\rangle^{\otimes n}$, we produce an output state of Schmidt rank at most L, but $n/2$ Bell pairs require a Schmidt rank of at least $2^{n/2}$ [7], which is a contradiction. □

As with the previous theorem, the proof of Theorem 5.3.2 also extends to the case of *approximating* a unitary easily; the fidelity between the closest $poly(n)$-Schmidt rank state and the $n/2$-Bell-pairs state decays exponentially in n. This implies that the operator distance (and diamond-norm distance) between U and any polynomial sum approximating U also becomes maximal in n.

5.4 Generalizations to other schemes

Our results have bounded how any locally acting unitary-based circuit cutting schemes can perform, but we have said relatively little about how a general scheme (one which can express circuits as the sum of other circuits of any form) may perform. It is therefore important to determine how broadly our results apply. In this section, we generalize our framework to encompass other locally acting circuit cutting schemes and discuss how apparently promising routes to generalize to non-local circuit cutting schemes do not work out.

To generalize our technique to other locally acting circuit cutting schemes note that the choice to decompose a unitary into other unitaries, while useful for illustration, was not maximally general. Instead, we can consider decomposing the *channel* associated to that unitary, $\mathcal{U}$, into other channels:

$$\mathcal{U} = \sum_{i}^{L} \alpha_i \mathcal{C}_i \tag{5.5}$$

$\mathcal{C}_i$ now respect an analogous cut locality condition. If a scheme obeys this locality condition (as [5] does) then the gadget can be applied to convert a connected circuit into a tensor product, allowing for classical simulation and extending Theorem 5.3.1 to this case. Even classical augmentation of the channel (e.g. with classical communication [7, 9]) would not break this simulability argument, further extending our results to this case.

It is less clear how the circuit cutting schemes that cut qubits time-wise (i.e. decompose identity channels [4, 9]) fit into this framework. The time-like circuit cutting schemes can be used to create partitioned blocks by cutting qubits that appear in two otherwise disconnected blocks. The choice of which qubits to cut is not immediately clear in our problem (which is to reduce the hardware requirements by just one qubit), instead we must try and find the analogous problem. If we only allow modifications outside the quantum comb, but still require blocks of at most $(n-1)$-qubits then the only option is to decompose local channels on the 2^{nd} qubit. In this case, our results extend.

Extending these results even further, to non-local (i.e. unrestricted) circuit cutting schemes, generally becomes much more challenging. The question now runs into issues of deciding the minimum circuit size necessary to implement a given function, related to the famously opaque minimum circuit size problem and its quantum analog [90]. Fortunately, existing schemes operate using only local cuts, making this question less relevant.

Finally, we wish to address an ostensible link between bounds on non-

local circuit cutting and the one clean qubit model [91]. The one clean qubit model is a restricted computational model consisting of an arbitrary circuit taking input of one clean qubit in some fiducial state and $(n-1)$ maximally mixed qubits. If one applies the types of circuit cutting methods discussed in this chapter to achieve an $(1, n-1)$ partition, one may come to the conclusion that the $(n-1)$-qubit computations will be acting on the maximally mixed states, which is classically simulatable. In this case, if the circuit cutting results in just polynomially many terms, the entire circuit cutting computation would be weakly simulatable (we can sample the output of the circuit), which would collapse the polynomial hierarchy, **PH**=**AM** [92], which is widely believed not to hold. This would constitute a rather elegant general no-go result for circuit cutting methods achieving a sub-exponential number of terms. However, the argument fails as circuit cutting does not necessarily apply a sum of just unitary channels to the maximally mixed input (e.g. in [3] different unitaries might be multiplied to the left and right side of the state, which is not a unitary channel and doesn't preserve the classical simulability of the maximally mixed state) or necessarily compute a $(n-1)$-qubit circuit on a subsystem of just the original (maximally mixed) input. Indeed this argument can be modified to show a slightly more general result: that all circuit cutting schemes must apply non-unital channels (which contain unitary channels), regardless of the number of terms generated (exponential or otherwise).

Corollary 5.4.1
No circuit cutting scheme can decompose any given unitary, U, on a given partition into a finite sum of only unital channels.

The proof (provided in the supplementary material) functions by using two SWAP gates to swap a clean qubit into a maximally mixed block.

5.5 Conclusion

In this chapter we have analysed the limits of locally acting circuit cutting schemes' ability to partition whole qubits. We found that for polynomially many two-qubit gates between the single qubit and the rest of the circuit, no locally acting scheme can achieve polynomial efficiency. We discussed how these results can be extended into other locally acting circuit cutting schemes, such as the superoperator or tensor network formalisms and suggested that they may apply to all local schemes.

This chapter suggests a clear future research direction: to either generalize these results to non-local circuit cutting schemes, or to attempt to

utilise some of the intuitions generated here to produce an efficient scheme for removing a single qubit from a circuit.

CHAPTER 6

Shadows of quantum machine learning

6.1 Introduction

Quantum machine learning is a rapidly growing field [93–95] driven by its potential to achieve quantum advantages in practical applications. A particularly interesting approach to make quantum machine learning applicable in the near term is to develop learning models based on parametrized quantum circuits [96–98]. Indeed, such quantum models have already been shown to achieve good learning performance in benchmarking tasks, both in numerical simulations [99–103] and on actual quantum hardware [104–107]. Moreover, based on widely-believed cryptography assumptions, these models also hold the promise to solve certain learning tasks that are intractable for classical algorithms [108, 109], including predicting ground state properties of highly-interacting quantum systems [110].

Despite these advances, quantum machine learning is facing a major obstacle for its use in practice. A typical workflow of a machine learning model involved, e.g., in driving autonomous vehicles, is divided into: (i) a *training phase*, where the model is trained, typically using training data or by reinforcement; followed by (ii) a *deployment phase*, where the trained model is evaluated on new input data. For quantum machine learning models, both of these phases require access to a quantum computer. But

given that in many practical machine learning applications, the trained model is meant for a widespread deployment, the current scarcity of quantum computing access dramatically reduces the applicability of quantum machine learning. One way of addressing this problem is by generating *shadow models* out of quantum machine learning models. That is, we propose inserting a *shadowing phase* between the training and deployment, where a quantum computer is used to collect information on the quantum model. Then a classical computer can use this information to evaluate the model on new data during the deployment phase.

The conceptual idea of generating shadows of quantum models was already proposed by Schreiber *et al.* [111], albeit under the terminology of *classical surrogates*. In that work, as well as in that of Landman *et al.* [112], the authors make use of the general expression of quantum models as trigonometric polynomials [113] to learn the Fourier representation of trained models and evaluate them classically on new data. However, these works also suggest that a classical model could potentially be trained directly on the training data and achieve the same performance as the shadow model, thus circumventing the need for a quantum model in the first place. This raises the concern that all quantum models that are compatible with a classical deployment would also lose all quantum advantage, hence severely limiting the prospects for a widespread use of quantum machine learning.
Therefore, two natural open questions are raised:

1. *Can shadow models achieve a quantum advantage over entirely classical (classically trained and classically evaluated) models?*

2. *Do there exist quantum models that do not admit efficiently evaluatable shadow models?*

In this chapter, we resolve both of these key open questions. We propose a general definition for shadow models, rooted in the fundamental idea that quantum machine learning models can be universally expressed as linear models [114]. This formulation of shadow models allows us to leverage various results and techniques from quantum information theory for the analysis of this model class. From a practical perspective, employing shadow tomography techniques [115–118] allows to easily construct diverse shadow models that will resonate with the practitioners of quantum machine learning. Furthermore, in our exploration of the computational capabilities of shadow models, we find them to capture a distinct computational class. Specifically, we demonstrate that, under widely-believed cryptography assumptions, there exist learning tasks where

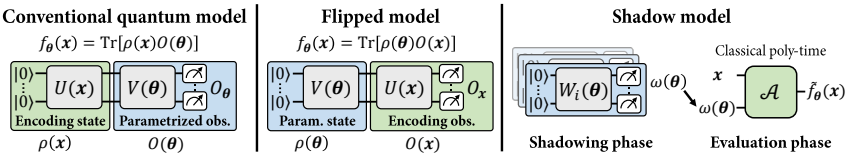


Figure 6.1: Quantum and shadow models. (left) Conventional quantum models can be expressed as inner products between a data-encoding quantum state $\rho(\boldsymbol{x})$ and a parametrized observable $O(\boldsymbol{\theta})$. The resulting linear model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$ naturally corresponds to a quantum computation, depicted here. (middle) We define flipped models $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ as quantum linear models where the role of the quantum state $\rho(\boldsymbol{\theta})$ and the observable $O(\boldsymbol{x})$ is flipped compared to conventional models. (right) Flipped models are associated to natural shadow models: one can use techniques from shadow tomography to construct a classical representation $\hat{\rho}(\boldsymbol{\theta})$ of the parametrized state $\rho(\boldsymbol{\theta})$ (during the shadowing phase), such that, for encoding observables $O(\boldsymbol{x})$ that are classically representable (e.g., linear combinations of Pauli observables), $\hat{\rho}(\boldsymbol{\theta})$ can be used by a classical algorithm to evaluate the model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ on new input data (during the evaluation phase). More generally, a shadow model is defined by (i) a shadowing phase where a (bit-string) advice $\omega(\boldsymbol{\theta})$ is generated by the evaluation of multiple quantum circuits $W_1(\boldsymbol{\theta}), \ldots, W_M(\boldsymbol{\theta})$, and (ii) an evaluation phase where this advice is used by a classical algorithm $\mathcal{A}$, along with new input data $\boldsymbol{x}$ to evaluate their labels $\widetilde{f_{\boldsymbol{\theta}}}(\boldsymbol{x})$. In section 6.3, we show that under this general definition, all shadow models are shadows of flipped models.

shadow models exhibit a provable quantum advantage over fully classical models. However, contrary to this advantage, we also establish that there exist quantum models that are strictly more powerful than the class of shadow models, based on common assumptions in complexity theory.

For ease of exposition, we will first adhere to a working definition of a shadow model as a model that is trained on a quantum computer, but can be evaluated classically on new input data with the help of information generated by a quantum computer (i.e., quantum-generated advice) that is independent of the new data. We will (informally) call a model "shadowfiable" if there exists a method of turning it into a shadow model. In Section 6.3, we will make our definitions more precise.

6.2 The flipped model

The construction of our shadow models starts from a simple yet key observation: all standard quantum machine learning models for supervised learning can be expressed as linear models [114]. To delve into this claim, we first draw upon early works that utilized parametrized quantum circuits in machine learning [99, 104]. These works proposed quantum models that are naturally expressed as linear functions of the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})] \tag{6.1}$$

where $\rho(\boldsymbol{x})$ are quantum states that encode classical data $\boldsymbol{x} \in \mathcal{X}$ and $O(\boldsymbol{\theta})$ are parametrized observables whose inner product with $\rho(\boldsymbol{x})$ defines $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ (see Fig. 6.1). In a regression task, one would use such a model to assign a real-valued label to an input $\boldsymbol{x}$, while in classification tasks, one would additionally apply, e.g., a sign function, to discretize its output into a class. From a circuit picture, such models can be evaluated on a quantum computer by: (i) preparing an initial state ρ_0, e.g., $|0\rangle\langle 0|^{\otimes n}$, (ii) evolving it under a data-dependent circuit $U(\boldsymbol{x})$, (iii) followed by a variational circuit $V(\boldsymbol{\theta})$, (iv) before finally measuring the expectation value of a Hermitian observable O. Together, steps (i) and (ii) define

$$\rho(\boldsymbol{x}) = U(\boldsymbol{x})\rho_0 U^\dagger(\boldsymbol{x}), \tag{6.2}$$

while steps (iii) and (iv) define

$$O(\boldsymbol{\theta}) = V^\dagger(\boldsymbol{\theta})OV(\boldsymbol{\theta}). \tag{6.3}$$

Since the early works, it is known that quantum linear models also capture quantum kernel models as a special case [119], simply by making $O(\boldsymbol{\theta})$ directly dependent on the training data of the learning task. Perhaps more surprisingly, quantum linear models can also encompass more general data re-uploading models, composed of several layers of data encoding and variational processing $U_1(\boldsymbol{x})V_1(\boldsymbol{\theta})U_2(\boldsymbol{x})\ldots$ Indeed, data re-uploading models can be mapped to linear models through circuit transformations (e.g., gate teleportation) that relocate all data-encoding gates to the first layer of the circuit [114].

6.2.1 Flipped model definition

The definition of a quantum linear model in Equation 6.1 can in general accommodate any pair of Hermitian operators in place of $\rho(\boldsymbol{x}), O(\boldsymbol{\theta})$. However, due to how these models are evaluated on a quantum computer, one commonly works under the constraint that $\rho(\boldsymbol{x})$ defines a quantum state (i.e., a positive semi-definite operator with unit trace). Indeed, from an operational perspective, $\rho(\boldsymbol{x})$ must be physically prepared on a quantum device before being measured with respect to the observable $O(\boldsymbol{\theta})$ (which only needs to be Hermitian in order to be a valid observable).

For reasons that will become clearer from the shadowing perspective, we define a so-called *flipped model*, where we flip the role of $\rho(\boldsymbol{x})$ and $O(\boldsymbol{\theta})$. That is, we consider

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})] \tag{6.4}$$

where $\rho(\boldsymbol{\theta})$ is a parametrized quantum state and $O(\boldsymbol{x})$ is an observable that encodes the data and can take more general forms than Equation 6.3 as we will see next. This model also corresponds to a straightforward quantum computation as $\rho(\boldsymbol{\theta})$ can be physically prepared before being measured with respect to $O(\boldsymbol{x})$.

A simple example of flipped model is for instance defined by:

$$\rho(\boldsymbol{\theta}) = V(\boldsymbol{\theta})\rho_0 V^\dagger(\boldsymbol{\theta}) \quad \& \quad O(\boldsymbol{x}) = \sum_{j=1}^{m} w_j(\boldsymbol{x})P_j \tag{6.5}$$

for an initial state ρ_0, a variational circuit $V(\boldsymbol{\theta})$, and a collection of Pauli observables $\{P_j\}_{j=1}^m$ weighted by data-dependent weights $w_j(\boldsymbol{x}) \in \mathbb{R}$. One can evaluate this model by repeatedly preparing $\rho(\boldsymbol{\theta})$ on a quantum computer, measuring it in a Pauli basis specified by a P_j, and weighting the outcome by $w_j(\boldsymbol{x})$. For other examples of flipped models, see Appendix 6.A.2.

As opposed to conventional quantum linear models, flipped models are well-suited to construct shadow models. Since the variational operators $\rho(\boldsymbol{\theta})$ are quantum states, one can straightforwardly use techniques from shadow tomography [115] to construct classical shadows $\hat{\rho}(\boldsymbol{\theta})$ of these states. What we call classical shadows $\hat{\rho}(\boldsymbol{\theta})$ here are collections of measurement outcomes obtained from copies of $\rho(\boldsymbol{\theta})$ that can be used to classically approximate expectation values of certain observables O (for a certain restricted family). If we take these observables to be our data-dependent $O(\boldsymbol{x})$, then we end up with a classical model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ that approximates our flipped model. Note here that one has total freedom on the classical shadow techniques they may use to define their shadow models, and a plethora of protocols have already been proposed in the literature [115–118]. But it is important to keep in mind that each of these protocols comes with its limitations, as it may restrict the class of states $\rho(\boldsymbol{\theta})$ or the class of observables $O(\boldsymbol{x})$ for which an *efficient and faithful* shadow model can be constructed. By *efficient* we refer here to the number of measurements performed on $\rho(\boldsymbol{\theta})$ and the time complexity of estimating the expectation values of observables $O(\boldsymbol{x})$ from these measurements. And by *faithful* we refer to the approximation error between the shadow model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ resulting from the shadow protocol and the original flipped model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$. For instance, in the example of Equation 6.5, we know that if all Pauli operators $\{P_j\}_{j=1}^m$ are k-local, then $\widetilde{\mathcal{O}}\left(3^k B^2 \varepsilon^{-2}\right)$ measurements of $\rho(\boldsymbol{\theta})$, where $B = \max_{\boldsymbol{x}} \sum_{j=1}^m |w_i(\boldsymbol{x})|$, are sufficient to guarantee $\max_{\boldsymbol{x}} \left|\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right| \leq \varepsilon$ with high probability. But for non-local Pauli operators (i.e., large k), this protocol becomes highly inefficient if we want to guarantee a low error ε.

Importantly, shadowfied flipped models are not limited to constructions based on classical shadow protocols. Given that the states $\rho(\boldsymbol{\theta})$ are not given to us a black-box (as is generally assumed in shadow tomography), one can use prior knowledge on these states to construct efficient shadowing procedure. For instance, if $\rho(\boldsymbol{\theta})$ is known to be a superposition of a tractable number of computational basis states, or well-approximated by a matrix product state (MPS) with low bond dimension, then efficient tomography protocols may be used [120].

6.2.2 Properties of flipped models

Flipped models are a stepping stone toward the claims of quantum advantage and "shadowfiability" that are the focus of this chapter. Nonetheless, they constitute a newly introduced model, which is why it is useful to understand first how they relate to previous quantum models and what

learning guarantees they can have.

Since conventional linear models of the form of Equation 6.1 play a central role in quantum machine learning, we start by asking the question: when can these models be represented by (efficiently evaluatable) flipped models? That is, given a conventional model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$, can we construct a flipped model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})]$ such that $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx f_{\boldsymbol{\theta}}(\boldsymbol{x}), \forall \boldsymbol{x}, \boldsymbol{\theta}$, and $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is as efficient to evaluate as $f_{\boldsymbol{\theta}}(\boldsymbol{x})$. Clearly, a conventional model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for which the parametrized operator $O(\boldsymbol{\theta})$ is also a quantum state (i.e., a positive semi-definite trace-1 operator) is by definition also a flipped model. Therefore, a natural strategy to flip a conventional model is to transform its observable $O(\boldsymbol{\theta})$ into a quantum state $\rho'(\boldsymbol{\theta})$. This transformation involves dealing with the negative eigenvalues of $O(\boldsymbol{\theta})$, which is straightforward[1], as well as *normalizing* these eigenvalues, which more importantly affects the efficiency of evaluating the resulting flipped model. Indeed, the normalization factor α that results from normalizing $O(\boldsymbol{\theta})$ corresponds to its trace norm $\|O\|_1 = \mathrm{Tr}\left[\sqrt{O^2}\right]$ and needs to be absorbed into the observable $O'(\boldsymbol{x}) = \alpha\rho(\boldsymbol{x})$ of the flipped model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ to guarantee $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$. This directly impacts the spectral norm $\|O'\|_\infty = \max_{|\psi\rangle} \langle O' \rangle_\psi = \alpha$ of the flipped model, and therefore the efficiency of its evaluation, as $\mathcal{O}(\|O'\|_\infty^2/\varepsilon^2)$ measurements of $\rho'(\boldsymbol{\theta})$ are needed in order to estimate $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ to additive error ε (see Appendix 6.B.1 for a derivation). Therefore, we end up showing that, for a conventional model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ acting on n qubits and with a bounded observable trace norm $\|O\|_1 \leq \alpha$, we can construct a flipped model acting on $m = n + 1$ qubits and with observable spectral norm $\|O'\|_\infty = \alpha$.

Interestingly, in the relevant regime where the number of qubits n, m used by the linear models involved in this flipping is logarithmic in $\|O\|_1$ (e.g., where O is a Pauli observable and hence $\|O\|_1 = 2^n$), we find that this requirement on the spectral norm $\|O'\|_\infty$ of the resulting flipped model is unavoidable in the worst case, up to a logarithmic factor in $\|O\|_1$. We refer to Appendix 6.B.3 for proof of these statements and a more in-depth discussion.

Another property of interest in machine learning is the generalization performance of a learning model. That is, we want to bound the gap between the performance of the model on its training set (so-called training error) and its performance on the rest of the data space (or expected error). Such bounds have for instance been derived in terms of the number of

[1] The sign of the eigenvalues of the observable $O(\boldsymbol{\theta})$ can be taken into account using an auxiliary qubit, without overheads in the efficiency of evaluation. See Appendix 6.B.3 for more details.

encoding gates in the quantum model [121], or the rank of its observable [122]. In the case of flipped model, we find instead a bound in terms of the number of qubits n and the spectral norm $\|O\|_\infty$ of the observable. Since these quantities are operationally meaningful, this gives us a natural way of controlling the generalization performance of our flipped models. Stated informally, we find that if a flipped model achieves a small error $|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \eta$ for all $\boldsymbol{x}$ in a training set of size M, then we only need M to scale as $\widetilde{\Omega}\left(\frac{n\|O\|_\infty^2}{\varepsilon\eta^2}\right)$ in order to guarantee a small expected error $|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq 2\eta$ with probability $1-\varepsilon$ over the entire data distribution.

Note that the dependence on n and $\|O\|_\infty$ is linear and quadratic, respectively, which means that we can afford a large number of qubits and a large spectral norm and still guarantee a good generalization performance. This is particularly relevant as the spectral norm is a controllable quantity, meaning we can easily fine-tune our models to perform well in training and generalize well. E.g., in the case of the model in Equation 6.5, this spectral norm is bounded by $\max_{\boldsymbol{x}} \sum_{j=1}^{m} |w_i(\boldsymbol{x})|$, which scales favorably with the number of qubits n if $m \in \mathcal{O}(\text{poly}(n))$ or if the vector $\boldsymbol{w}(\boldsymbol{x})$ is sparse.

6.2.3 Quantum advantage of a shadow model

We recall that we (informally) define shadow models as models that are trained on a quantum computer, but, after a shadowing procedure that collects information on the trained model, are evaluated classically on new input data. In this section, we consider the question of achieving a quantum advantage using such shadow models. It may seem at first sight that this question has a straightforward answer, which is "no": if the function learned by a model is classically computable, then there should be no room for a quantum advantage. However, as demonstrated in Refs. [109, 123], one can also achieve a quantum advantage based on so-called *trap-door functions*. These are functions that are believed to be hard to compute classically, unless given a key (or advice) that allows for an efficient classical computation. Notably, there exist trap-door functions where this key can be efficiently computed using a quantum computer, but not classically. This allows us to construct shadow models that make use of this quantum-generated key to compute an otherwise classically untractable function.

Similarly to related results showing a quantum advantage in machine learning with classical data [108, 124], we consider a learning task where the target function (i.e., the function generating the training data) is derived from cryptographic functions that are widely believed to be

hard to compute classically. More precisely, we introduce a variant of the discrete cube root learning task [109], which is hard to solve classically under a hardness assumption related to that of the RSA cryptosystem [125]. In this task, we consider target functions defined on $\mathbb{Z}_N = \{0, \ldots, N-1\}$ as

$$g_s(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \sqrt[3]{\boldsymbol{x}} \bmod N \in [s, s + \frac{N-1}{2}], \\ 0, & \text{otherwise} \end{cases} \tag{6.6}$$

where $N = pq$ is an n-bit integer, product of two primes $p,\ q$ of the form $3k+2,\ 3k'+2$, such that the discrete cube root is properly defined as the inverse of the function $\boldsymbol{y}^3 \bmod N$. These target functions are particularly appealing because of a number of interesting properties:

(i) It is believed that given only $\boldsymbol{x}$ and N as input, computing $g(\boldsymbol{x}) = \sqrt[3]{\boldsymbol{x}} \bmod N$ with high probability of success over random draws of $\boldsymbol{x}$ and N is classically intractable. This assumption is known as the discrete cube root (DCR) assumption.

(ii) On the other hand, computing $\boldsymbol{x}^a \bmod N$ is classically efficient for any $a \in \mathbb{Z}_N$. For $a = 3$, this implies that $g^{-1}(\boldsymbol{y}) = \boldsymbol{y}^3 \bmod N$ is a one-way function, under the DCR assumption.

(iii) The function $g(\boldsymbol{x}) = \sqrt[3]{\boldsymbol{x}} \bmod N$ has a "trap-door", in that there exists another way of computing it efficiently. For every N (as specified above), there exists a *key* $d \in \mathbb{Z}_N$ such that $g(\boldsymbol{x}) = \boldsymbol{x}^d \bmod N$. Finding d is efficient quantumly by using Shor's factoring algorithm [126], but hard classically under the DCR assumption.

Observations (i) and (ii) can be leveraged to show that learning the functions g_s from examples is also intractable. Indeed, Alexi *et al.* [127] showed that a classical algorithm that could faithfully capture a single bit $g_s(\boldsymbol{x})$ of the discrete cube root of $\boldsymbol{x}$, for even a $1/2 + 1/\mathrm{poly}(n)$ fraction of all $\boldsymbol{x} \in \mathbb{Z}_N$, could also be used to reconstruct $g(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{Z}_N$, with high probability of success. Since, from observation (ii), the training data for the learning algorithm can also be generated efficiently classically from N, a classical learner that learns $g_s(\boldsymbol{x})$ correctly for a $1/2 + 1/\mathrm{poly}(n)$ fraction of all $\boldsymbol{x} \in \mathbb{Z}_N$ would then contradict the DCR assumption.

Observation (iii) allows us to define the following flipped model:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta}) O(\boldsymbol{x})] \tag{6.7}$$
$$\rho(\boldsymbol{\theta}) = |d', s'\rangle\langle d', s'| \ \ \& \ O(\boldsymbol{x}) = \sum_{d', s'} \widehat{g}_{d', s'}(\boldsymbol{x}) \, |d', s'\rangle\langle d', s'| \, .$$

That is, $\rho(\boldsymbol{\theta})$ (for $\boldsymbol{\theta} = (N, s')$) specifies candidates for the key d' and the parameter s' of interest, while $O(\boldsymbol{x})$ uses that information to compute

$$\widehat{g}_{d',s'}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x}^{d'} \bmod N \in [s', s' + \frac{N-1}{2}], \\ 0, & \text{otherwise.} \end{cases} \tag{6.8}$$

The state $\rho(\boldsymbol{\theta}) = |d, s'\rangle\langle d, s'|$ for the right key d can be prepared efficiently using Shor's algorithm applied on N (provided with the training data). As for $O(\boldsymbol{x})$, it simply processes classically a bit-string to compute $\widehat{g}_{d',s'}(\boldsymbol{x})$ efficiently, which corresponds to $g_s(\boldsymbol{x})$ when $(d', s') = (d, s)$. Finding an s' close to s is an easy task given training data and $d' = d$. Since $\rho(\boldsymbol{\theta})$ is a computational basis state, this flipped model admits a trivial shadow model where a single computational basis measurement of $\rho(\boldsymbol{\theta})$ allows to evaluate $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ classically for all $\boldsymbol{x}$. Therefore, we end up showing the following theorem:

Theorem 6.1 (Quantum advantage (informal))
There exists a learning task where a shadow model first trained using a quantum computer then evaluated classically on new input data, can achieve an arbitrarily good learning performance, while any fully classical model cannot do significantly better than random guessing, under the hardness of classically computing the discrete cube root.

In Appendix 6.C we formalize the statement of this result using the PAC framework and provide more details on the setting and the proofs.

6.3 General shadow models

As mentioned at the start of this chapter, shadow models are not limited to shadowfied flipped models, and the main alternative proposals are based on the Fourier representation of quantum models [111, 112]. It is clear that Fourier models are defined very differently from flipped models, but one may wonder whether they nonetheless include shadowfied flipped models as a special case, or the other way around.

In this section, we first start by showing that there exist quantum models that admit shadow models (i.e., are shadowfiable) but cannot be shadowfied efficiently using a Fourier approach. This then motivates our proposal for a general definition of shadow models, and we show that, under this definition, all shadow models can be expressed as shadowfied flipped models. Finally, we show the existence of quantum models that are not shadowfiable at all under likely complexity theory assumptions.

6.3.1 Shadow models beyond Fourier

An interesting approach to construct shadows of quantum models is based on their natural Fourier representation. It has been shown [113, 128] that quantum models can be expressed as generalized Fourier series of the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{\boldsymbol{\omega}\in\Omega} c_{\boldsymbol{\omega}}(\boldsymbol{\theta}) e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}} \tag{6.9}$$

where the accessible frequencies Ω only depend on properties of the encoding gates used by the model (notably the number of encoding gates and their eigenvalues). Since these frequencies can easily be read out from the circuit, one can proceed to form a shadow model by estimating their associated coefficients $c_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ using queries of the quantum model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ at different values $\boldsymbol{x}$ and, e.g., a Fourier transform [111]. Given a good approximation of these coefficients, one can then compute estimates of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for arbitrary new inputs $\boldsymbol{x}$. We will refer to such a shadowing approach that considers the quantum model as a black-box, aside from the knowledge of its Fourier spectrum, as the Fourier shadowing approach.

Although we will be explicit about this in the next subsection, we will consider a shadowing procedure to be successful, if, with high probability, the resulting shadow model agrees with the original model on all inputs[2], i.e.,

$$\max_{\boldsymbol{x}\in\mathcal{X}} \left| f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right| \leq \varepsilon, \tag{6.10}$$

for a specified $\varepsilon \geq 0$.

We show that the Fourier shadowing approach can suffer from an exponential sample complexity in the dimension of the input data $\boldsymbol{x}$, making it intractable for high-dimensional input spaces. To see this, consider the linear model:

$$\begin{gathered} f_{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x}) O(\boldsymbol{y})] \\ \rho(\boldsymbol{x}) = \bigotimes_{i=1}^{n} R_Y(x_i) \left|0\right\rangle\!\left\langle 0\right| R_Y^\dagger(x_i) \;\&\; O(\boldsymbol{y}) = \left|\boldsymbol{y}\right\rangle\!\left\langle \boldsymbol{y}\right| . \end{gathered} \tag{6.11}$$

for $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \{0,1\}^{\otimes n}$. Let us first restrict our attention to the domain $\boldsymbol{x} \in \{0,\pi\}^n$. It is quite clear that on this domain, $f_{\boldsymbol{y}}(\boldsymbol{x}) = \delta_{\boldsymbol{x}/\pi,\boldsymbol{y}}$ plays the role of a database search oracle, where the database has 2^n

[2] We want the shadowing procedure to be successful independently of the data distribution under which the model should be trained, which justifies this definition. We discuss this point further in Appendix 6.D.4.

elements and a unique marked element $\boldsymbol{y}$. From lower bounds on database search, we know that $\Omega(2^n)$ calls to this oracle are needed to find $\boldsymbol{y}$ [129]. This implies that a Fourier shadowing approach would require $\Omega(2^n)$ calls to $f_{\boldsymbol{y}}(\boldsymbol{x}) = \delta_{\boldsymbol{x}/\pi,\boldsymbol{y}}$ in order to guarantee $\max_{\boldsymbol{x}\in\mathcal{X}} \left|\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right| \leq 1/4$. In Appendix 6.D.3, we explain how this result can be generalized to the full domain $\boldsymbol{x} \in \mathbb{R}^n$, and we relate this bound on the sample complexity to the Fourier decomposition of the model.

On the other hand, note that the flipped model associated to $f_{\boldsymbol{y}}(\boldsymbol{x})$ allows for a straightforward shadowing procedure. Indeed, by preparing $O(\boldsymbol{y})$ and measuring it in the computational basis, one straightforwardly obtains $\boldsymbol{y}$ and can therefore classically compute the expectation value of any tensor product observable $\rho(\boldsymbol{x})$ as specified by Equation 6.11. Therefore, we have shown that there exist shadowfiable models that are not efficiently Fourier-shadowfiable, i.e., for which a shadowing procedure based solely on the knowledge of their Fourier spectrum and on black-box queries has query complexity that is exponential in the input dimension.

6.3.2 All shadow models are shadows of flipped models

We give a general definition of shadow models that can encompass all methods that have been proposed to generate them. In contrast to the definition of classical surrogates proposed by Schreiber *et al.* [111], we give explicit definitions for the shadowing and evaluation phases of shadow models which makes explicit the need for a quantum computer in the shadowing phase. Indeed, as mentioned in the introduction, the term *classical surrogate* has been used to describe both a classically evaluatable model obtained from a quantum shadowing procedure and a fully classical model trained directly on the data. We want to avoid this confusion in the definition of shadow models. We view a general shadowing phase as the generation of *advice* that can be used to classically evaluate a quantum model. This advice is generated by the execution of quantum circuits that may or may not depend on the (trained) quantum circuit from the training phase. For instance, when we shadowfy a flipped model, we simply prepare the parametrized states $\rho(\boldsymbol{\theta})$ and use (randomized) measurements to generate an operationally meaningful classical description. In the case of Fourier shadowing, this advice is instead generated by evaluations of the quantum model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for different inputs $\boldsymbol{x} \in \mathbb{R}^d$ that are rich enough to learn the Fourier coefficients of this model. We propose the following definition:

Definition 6.3.1 (General shadow model)
Let $W_1(\boldsymbol{\theta}), ..., W_M(\boldsymbol{\theta})$ be a sequence of $\mathcal{O}(poly(m))$-time quantum circuits

applied on all-zero states $|0\rangle^{\otimes m}$*, and that can potentially be chosen adaptively. Call* $\omega(\boldsymbol{\theta}) = (\omega_1(\boldsymbol{\theta}), \ldots, \omega_M(\boldsymbol{\theta}))$ *the outcomes of measuring the output states of these circuits in the computational basis. A general shadow model is defined as:*

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta})) \tag{6.12}$$

where $\mathcal{A}$ *is a classical* $\mathcal{O}(poly(M, m, d))$*-time algorithm that processes the outcomes* $\omega(\boldsymbol{\theta})$ *along with an input* $\boldsymbol{x} \in \mathbb{R}^d$ *to return the (real-valued) label* $f_{\boldsymbol{\theta}}(\boldsymbol{x})$*.*

From this definition, a shadow model is a classically evaluatable model that uses quantum-generated advice. Crucially, this advice must be independent of the data points $\boldsymbol{x}$ we wish to evaluate the model on in the future. We distinguish the notion of a shadow model from that of a *shadowfiable* quantum model, that is a quantum model that admits a shadow model:

Definition 6.3.2 (Shadowfiable model)
A model $f_{\boldsymbol{\theta}}$ *acting on* n *qubits is said to be shadowfiable if, for* $\varepsilon, \delta > 0$*, there exists a shadow model* $\widetilde{f}_{\boldsymbol{\theta}}$ *such that, with probability* $1 - \delta$ *over the quantum generation of the advice* $\omega(\boldsymbol{\theta})$ *(i.e., the shadowing phase), the shadow model satisfies*[2]

$$\max_{\boldsymbol{x} \in \mathcal{X}} \left| f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right| \leq \varepsilon, \tag{6.13}$$

and uses $m, M \in \mathcal{O}(poly(n, 1/\varepsilon, 1/\delta))$ *qubits and circuits to generate its advice* $\omega(\boldsymbol{\theta})$*.*

While we have seen that there exist shadowfiable models that cannot be shadowfied efficiently using a Fourier approach, we show that all shadowfiable models as defined above can be approximated by shadowfiable flipped models.

Lemma 1 (Flipped models are shadow-universal)
All shadowfiable models as defined in Defs. 6.3.1 and 6.3.2 can be approximated by flipped models $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ *with the guarantee that computational basis measurements of* $\rho(\boldsymbol{\theta})$ *and efficient classical post-processing can be used to evaluate* $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ *to good precision with high probability.*

This result is essentially based on the observation that the evaluation of a general shadow model as defined in Def. 6.3.1 can be done entirely coherently. Instead of classically running the algorithm $\mathcal{A}$ using the random advice $\omega(\boldsymbol{\theta})$, one can quantumly simulate this algorithm (using

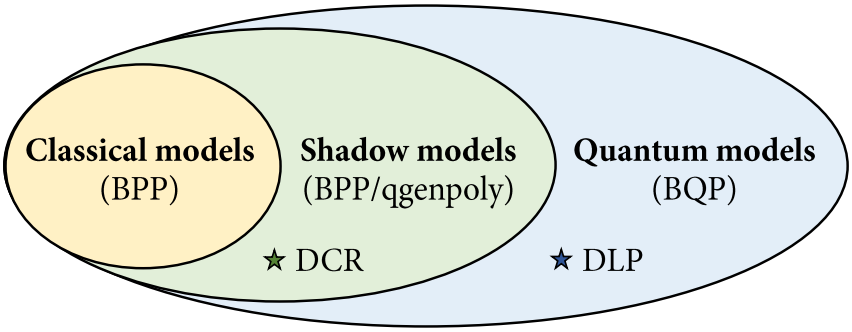


Figure 6.2: Separations between classical, shadow, and quantum models. Under the assumption that the discrete cube root (DCR) cannot be computed classically in polynomial time, we have a separation between shadow models (captured by the class BPP/qgenpoly) and classical models (in BPP). Under the assumption that there exist functions that can be computed in quantum polynomial time but not in classical polynomial time with the help of advice (i.e., BQP $\not\subset$ P/poly), we have a separation between quantum models (universal for BQP) and shadow models (BPP/qgenpoly). A candidate function for this separation is the discrete logarithm (DLP).

a reversible execution) and execute it on the coherent advice $\rho(\boldsymbol{\theta}) = |\omega(\boldsymbol{\theta})\rangle\langle\omega(\boldsymbol{\theta})|$ generated by $\{W_1(\boldsymbol{\theta}), ..., W_M(\boldsymbol{\theta})\}$ before the computational basis measurements. We refer to Appendix 6.D.1 for a more detailed statement and proof.

6.3.3 Not all quantum models are shadowfiable

From the discrete cube root learning task, we already understand that a learning separation can be established between classical and shadowfiable models. We would also like to understand whether a learning separation exists between shadowfiable models and general quantum models, or equivalently, whether all quantum models are shadowfiable. We show that this also is not the case, under widely believed assumptions (see Fig. 6.2).

Theorem 6.2 (Not all shadowfiable)
Under the assumption that BQP $\not\subset$ P/poly, there exist quantum models, i.e., models in BQP, that are not shadowfiable, i.e., that are not in BPP/qgenpoly.

We start by noting that shadow models can be characterized by a

complexity class we define as BPP/qgenpoly,[3] which contains all functions that can be computed efficiently classically with the help of polynomially-sized advice *generated efficiently by a quantum computer*. This class is trivially contained in the standard class BPP/poly, which doesn't have any constraint on how the advice is generated and can be derandomized to P/poly (i.e., BPP/poly=P/poly [130]). Note however that BPP/qgenpoly constitutes a physically relevant class, since it only contains problems that can be solved efficiently by classical and quantum computers, as opposed to P/poly, which contains undecidable problems, such as a version of the halting problem. We refer to Appendix 6.A.3 for formal definitions of these complexity classes, and an in-depth discussion.

On the other hand, it is easy to show that quantum models (more precisely quantum linear models) can also represent any function in BQP, i.e., all functions that are efficiently computable on a quantum computer. For this, one simply takes a simple encoding of an n-bit input $\boldsymbol{x}$:

$$\rho(\boldsymbol{x}) = \bigotimes_{i=1}^{n} X_i^{x_i} \left|0\right\rangle\!\left\langle 0\right| X_i^{x_i} \tag{6.14}$$

along with an observable

$$O_n = U_n^\dagger Z_1 U_n \tag{6.15}$$

specified by an arbitrary n-qubit circuit U_n in BQP and the Pauli-Z operator applied on it first qubit. The resulting model $f_n(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O_n]$ can then be used to decide any language in BQP.

Combining these two observations, we get that the proposition "all quantum models are shadowfiable" would imply that BQP $\subseteq$ BPP/qgenpoly $\subseteq$ P/poly, which violates the widely-believed conjecture [15] that BQP $\not\subseteq$ P/poly (see Appendix 6.D.2 for a formal proof). To give an example of candidates of non-shadowfiable quantum models, the discrete logarithm $\log_g x \bmod p$ (or even one bit of it) is provably in BQP but is not believed to be in P/poly. Therefore, a model that could be used to compute the discrete logarithm (e.g., the quantum model of Liu *et al.* [108]) is likely not shadowfiable.

[3] Stands for Bounded-error Probabilistic Polynomial-time with quantumly generated (polynomial-time) advice of polynomial size. In this chapter, we talk mostly about complexity classes for decision problems. Note however that for models, since these compute real-valued functions (represented to machine precision), we should instead consider the function-problem version of these complexity classes.

6.4 Discussion

In this work, we examined the class of quantumly trainable, classically evaluatable models we refer to as shadow models. Our analysis has shown that these models can be universally captured by a restricted family of quantum linear models, wherein data-encoding and variational operations are flipped compared to conventional quantum models. Furthermore, we demonstrated that shadow models belong to an intriguing complexity class, coined BPP/qgenpoly, exhibiting superiority over classical models (in BPP) but inferiority to fully quantum models (in BQP), based on prevalent complexity theory assumptions.

By presenting shadows models as flipped linear models, we illustrated how shadow tomography protocols could be applied straightforwardly to construct shadow models in practice. Yet, it is important to note a crucial distinction between a shadow tomography scenario and a shadow model: in the latter, one has control over the quantum state intended for shadowing. This distinction introduces new possibilities for devising 'state-aware' shadow tomography protocols aimed at constructing shadow models. This could potentially alleviate some of the limitations of current classical shadow protocols.

Considering our findings on learning separations, we identified a noteworthy characteristic of shadow models: their ability to quantumly compute useful advice for a classical evaluation algorithm, enabling them to tackle otherwise classically-intractable tasks. The example we presented, based on trap-door functions, readily allows for such constructions, but it remains somewhat contrived. Exploring similar constructions for physically-relevant problems, such as predicting ground state properties of complex quantum systems, would be an intriguing avenue for future research.

6.A Formal definitions

6.A.1 Linear models

Definition 6.A.1 (Conventional linear model)
Let $U(\boldsymbol{x})$ be an encoding quantum circuit that is parametrized by input data $\boldsymbol{x} \in \mathbb{R}^d$, ρ_0 a fixed input quantum state (diagonal in the computational basis), $V(\boldsymbol{\theta})$ a variational quantum circuit parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^p$ and $O = \sum_{i=1}^m w_i O_i$ an observable specified by a (trainable) linear combination of Hermitian matrices $\{O_i\}_{i=1}^m$. A conventional linear model is defined by the parametrized function:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x}) O(\boldsymbol{\theta})] \tag{6.16}$$

for $\rho(\boldsymbol{x}) = U(\boldsymbol{x})\rho_0 U^\dagger(\boldsymbol{x})$ and $O(\boldsymbol{\theta}) = V^\dagger(\boldsymbol{\theta}) O V(\boldsymbol{\theta})$ (when the weights $\{w_i\}_{i=1}^m$ are also trainable, we include them in the parameters $\boldsymbol{\theta}$ of the model).

Definition 6.A.2 (Flipped model)
Let $V(\boldsymbol{\theta})$ be a variational quantum circuit parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^p$, ρ_0 a fixed input quantum state (diagonal in the computational basis), $U(\boldsymbol{x})$ an encoding quantum circuit that is parametrized by input data $\boldsymbol{x} \in \mathbb{R}^d$ and $O_{\boldsymbol{x}} = \sum_{i=1}^m w(\boldsymbol{x})_i O_i$ an observable specified by a linear combination of Hermitian matrices $\{O_i\}_{i=1}^m$, weighted by a data-dependent function $w : \mathbb{R}^d \to \mathbb{R}^m$. A flipped model is defined by the parametrized function:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta}) O(\boldsymbol{x})] \tag{6.17}$$

for $\rho(\boldsymbol{\theta}) = V(\boldsymbol{\theta})\rho_0 V^\dagger(\boldsymbol{\theta})$ and $O(\boldsymbol{x}) = U^\dagger(\boldsymbol{x}) O_{\boldsymbol{x}} U(\boldsymbol{x})$.

6.A.2 Shadow models

Definition 6.A.3 (Shadow model)
Let $\{W_1(\boldsymbol{\theta}), ..., W_M(\boldsymbol{\theta})\}$ be a sequence of m-qubit unitary circuits that are dependent on a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$, and can potentially be chosen adaptively. We define the quantum-generated *advice $\omega(\boldsymbol{\theta}) = (\omega_1(\boldsymbol{\theta}), \ldots, \omega_M(\boldsymbol{\theta}))$ as the measurement outcomes $\omega_i(\boldsymbol{\theta})$ obtained by measuring the states $W_i(\boldsymbol{\theta}) |0\rangle^{\otimes m}$ in the computational basis (and a description of their associated circuits $W_i(\boldsymbol{\theta})$). A shadow model is defined as the parametrized function:*

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta})) \tag{6.18}$$

for $\mathcal{A}$ a classical $\mathcal{O}(poly(m, M, d))$-time algorithm that takes as input the advice $\omega(\boldsymbol{\theta})$, a data vector $\boldsymbol{x} \in \mathbb{R}^d$ and outputs a real-valued label $f_{\boldsymbol{\theta}}(\boldsymbol{x})$

Examples of shadow models:

- Take a flipped model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ for $\rho(\boldsymbol{\theta})$ a quantum state generated by a circuit $V(\boldsymbol{\theta})$ applied to $|0\rangle^{\otimes n}$ and $O(\boldsymbol{x}) = \sum_{i=1}^{m} w(\boldsymbol{x})_i P_i$ where $\{P_i\}_{i=1}^{m}$ are all k-local Pauli strings acting on n qubits.
  A simple shadow model associated to this flipped model consists in estimating all the expectation values $\langle P_i \rangle \approx \mathrm{Tr}[\rho(\boldsymbol{\theta})P_i]$ via repeated measurements of $\rho(\boldsymbol{\theta})$ in the eigenbasis of each of the $m = \binom{n}{k}3^k$ Pauli strings P_i, and taking their weighted combination $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{m} w(\boldsymbol{x})_i \langle P_i \rangle$. In this case, the unitary circuits $\{W_1(\boldsymbol{\theta}), ..., W_M(\boldsymbol{\theta})\}$ are simply obtained by $V(\boldsymbol{\theta})$ followed by a basis change unitary (corresponding to the Pauli basis of P_i). As for the classical algorithm $\mathcal{A}$, this is simply a collection of mean estimators that compute estimates $\langle P_i \rangle$ out of measurement outcomes, followed by the computation of a weighted sum. The number of measurements needed for this shadow model to guarantee $\left|\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right| \leq \varepsilon, \ \forall \boldsymbol{x} \in \mathbb{R}^d$ is $M \in \widetilde{\mathcal{O}}\left(\frac{m \max_{\boldsymbol{x}} \|w(\boldsymbol{x})\|_1^2}{\varepsilon^2}\right)$. Indeed, estimating each $\langle P_i \rangle$ to additive error $\frac{\varepsilon}{\max_{\boldsymbol{x}} \|w(\boldsymbol{x})\|_1}$ allows us to guarantee the desired total additive error, and each of this estimates can be obtained using $\widetilde{\mathcal{O}}\left(\frac{\max_{\boldsymbol{x}} \|w(\boldsymbol{x})\|_1^2}{\varepsilon^2}\right)$ samples.
  A more interesting shadow model relies on Pauli classical shadows [131] where random Pauli measurements are used to construct $\omega(\boldsymbol{\theta})$. Median-of-mean estimators then use these measurement outcomes to compute empirical estimates $\hat{\rho}(\boldsymbol{\theta})$ of $\rho(\boldsymbol{\theta})$ and approximate all expectation values $\mathrm{Tr}[\rho(\boldsymbol{\theta})P_i]$. The advantage of this shadow model is that it requires $M \in \widetilde{\mathcal{O}}\left(\frac{3^k \max_{\boldsymbol{x}} \|w(\boldsymbol{x})\|_1^2}{\varepsilon^2}\right)$ measurements for the same guarantees as the shadow model above, which constitutes savings of a factor $\widetilde{\mathcal{O}}\left(\binom{n}{k}\right)$.

- Another interesting flipped model that can be turned into a shadow model: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ for $\rho(\boldsymbol{\theta})$ arbitrarily defined and $O(\boldsymbol{x}) = |\psi(\boldsymbol{x})\rangle\langle\psi(\boldsymbol{x})|$, for some pure states $|\psi(\boldsymbol{x})\rangle$. Given that, $\mathrm{Tr}\left[O(\boldsymbol{x})^2\right] = 1, \ \forall \boldsymbol{x} \in \mathbb{R}^d$, then Clifford classical shadows [131] allow to construct a representation $\omega(\boldsymbol{\theta})$ of $\rho(\boldsymbol{\theta})$ that guarantees $\left|\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right| \leq \varepsilon, \ \forall \boldsymbol{x} \in \mathbb{R}^d$ using only $M \in \widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon^2}\right)$ measurements. However, for

this estimation to be computationally efficient, the states $|\psi(\boldsymbol{x})\rangle$ need to be stabilizer states, or be generated by few (i.e., $\mathcal{O}(\log(n))$) non-Clifford gates [132].

- Consider the model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{m} w(\boldsymbol{x})_i \operatorname{Tr}[\rho(\boldsymbol{\theta})P_i]^2$ for $m = 4^n$, i.e., $\{P_i\}_{i=1}^m$ are all n-qubit Pauli strings. By preparing two-copy states $\rho(\boldsymbol{\theta}) \otimes \rho(\boldsymbol{\theta})$ and performing simultaneous Bell measurements between pairs of qubits of these two copies, $M = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ such measurements give a $\omega(\boldsymbol{\theta})$ rich enough to compute any $\operatorname{Tr}[\rho(\boldsymbol{\theta})P_i]^2$ to precision ε [133]. Therefore, evaluating $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ requires only $M \in \widetilde{\mathcal{O}}\left(\frac{\max_{\boldsymbol{x}} \|w(\boldsymbol{x})\|_1^2}{\varepsilon^2}\right)$ measurements using this shadow model, compared to $2^{\Omega(n)}$ for a shadow model that would construct $\omega(\boldsymbol{\theta})$ using single-copy measurements only (assuming that for all $i \in \{1, \ldots, m\}$, there exists an $\boldsymbol{x} \in \mathbb{R}^d$ such that $w(\boldsymbol{x})_i \neq 0$). For a $w(\boldsymbol{x})$ that is k-sparse for all $\boldsymbol{x}$, with $k \ll m$, this constitutes an exponential separation.

6.A.3 Complexity classes

Definition 6.A.4 (BQP)
A language L is in BQP if and only if there exists a polynomial-time uniform family of quantum circuits $\{U_n : n \in \mathbb{N}\}$, such that

1. *For all $n \in \mathbb{N}$, U_n takes as input an n-qubit computational basis state, and outputs one bit obtained by measuring the first qubit in the computational basis.*
2. *For all $x \in L$, the probability that the output of $U_{|x|}$ applied on the input x is 1 is greater or equal to $2/3$.*
3. *For all $x \notin L$, the probability that the output of $U_{|x|}$ applied on the input x is 0 is greater or equal to $2/3$.*

Definition 6.A.5 (P/poly)
A language L is in P/poly if and only if there exists a polynomial-time classical algorithm $\mathcal{A}$ and a sequence of polynomial-size advice strings $\{\alpha_n \in \{0,1\}^{poly(n)}\}_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$ and all $x \in \{0,1\}^n$:

$$\mathcal{A}(x, \alpha_n) = 1 \iff x \in L. \tag{6.19}$$

Definition 6.A.6 (BPP/qgenpoly)
A language L is in BPP/qgenpoly if and only if there exists a polynomial-time uniform family of quantum circuits $\{U_n : n \in \mathbb{N}\}$ and a polynomial-time probabilistic classical algorithm $\mathcal{A}$, such that

1. *For all $n \in \mathbb{N}$, U_n takes as input the computational basis state $|0\rangle^{\otimes m}$ for $m \in \mathcal{O}(poly(n))$, and outputs m bits obtained by measuring all qubits in the computational basis. This constitutes the* quantum-generated *advice ω_n.*

2. *For all $x \in \{0,1\}^*$, $\mathcal{A}$ takes as input x and $\omega_{|x|}$.*

3. *For all $x \in L$, the probability that $\mathcal{A}(x, \omega_{|x|})$ outputs 1 is greater or equal to $2/3$, taken over the randomness of $\omega_{|x|}$ and the internal randomness of $\mathcal{A}$.*

4. *For all $x \notin L$, the probability that $\mathcal{A}(x, \omega_{|x|})$ outputs 0 is greater or equal to $2/3$, taken over the randomness of $\omega_{|x|}$ and the internal randomness of $\mathcal{A}$.*

The following inclusions are easy to show:

$$\mathsf{BPP} \overset{(i)}{\subseteq} \mathsf{BPP/qgenpoly} \overset{(ii)}{\subseteq} \mathsf{BQP}. \tag{6.20}$$

(i) follows from making the quantum-generated advice ω_n empty in the definition of BPP/qgenpoly. (ii) follows from the ability to efficiently simulate classical computations on a quantum computer. As illustrated in Figure 6.4, the algorithm $\mathcal{A}$ in the definition of BPP/qgenpoly can be simulated unitarily, and absorbed in the uniform family of quantum circuits $\{U_n : n \in \mathbb{N}\}$, resulting in polynomial-time quantum circuits that fit the definition of BQP.

Our learning separations results, i.e., Theorem 6.1 (Lemmas 6.C.3 and 6.C.4 in the Appendix) and Theorem 6.2 (Lemma 6.D.2 in the Appendix), can be seen as evidence that the inclusions (i) and (ii) are strict, based on complexity-theory assumptions. Other notable inclusions that are useful to prove our results are:

$$\mathsf{BPP/qgenpoly} \subsetneq \mathsf{BPP/poly} = \mathsf{P/poly}. \tag{6.21}$$

The equality BPP/poly = P/poly follows from the derandomization results of Adleman [130], which show that the random errors made by an algorithm in BPP/poly can be canceled by an appropriate choice of the random bits used by the randomized algorithm, which are then appended to the poly-sized advice to obtain an algorithm in P/poly. The inclusion BPP/qgenpoly $\subseteq$ BPP/poly simply comes from the fact that BPP/poly is not restricted in *how* the poly-sized advice is generated, and the remainder of its definition is identical to that of BPP/qgenpoly. The restriction we

make on *how* the advice is generated in BPP/qgenpoly makes it a physically relevant complexity class, as opposed to (BP)P/poly. All problems in BPP/qgenpoly can be solved efficiently (i.e., in polynomial time) using (classical and) quantum computers, while P/poly notably contains undecidable problems. Consider for instance the unary version of the halting problem: UHALT $= \{1^n, \text{where } n \text{ encodes } (M, x) \text{ such that the Turing machine } M \text{ halts on } x\}$ is an undecidable language (as any algorithm that would decide it would also be able to decide the traditional halting problem), but by considering the advice $\alpha_n = 1$ if $1^n \in$ UHALT and 0 otherwise (uniquely defined for each input size n), one trivially obtains an algorithm in P/poly that solves it. The fact that this advice cannot be generated by a uniform family of (poly-time) circuits is irrelevant for the class P/poly, which is at the source of this result. However it is relevant for the class BPP/qgenpoly, and, in fact, having UHALT $\in$ BPP/qgenpoly would mean that one could solve an undecidable problem using uniform (poly-time) circuits. The impossibility of this hypothetical result gives BPP/qgenpoly $\subsetneq$ P/poly.

6.B Properties of flipped models

6.B.1 Sample complexity of evaluating quantum models

Consider a linear quantum model (either conventional or flipped) of the form

$$f_{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})], \tag{6.22}$$

for a quantum state $\rho(\boldsymbol{x})$ parametrized by a vector $\boldsymbol{x} \in \mathbb{R}^d$ and $O(\boldsymbol{y})$ a Hermitian observable parametrized by a vector $\boldsymbol{y} \in \mathbb{R}^p$. Assume that we can prepare single copies of $\rho(\boldsymbol{x})$ and that we can measure them in the eigenbasis of $O(\boldsymbol{y})$. We ask: given error parameters $\varepsilon, \delta > 0$, how many such measurements of $\rho(\boldsymbol{x})$ do we need in order to compute an estimate $\widehat{f}_{\boldsymbol{y}}(\boldsymbol{x})$ of $f_{\boldsymbol{y}}(\boldsymbol{x})$ such that $\left|\widehat{f}_{\boldsymbol{y}}(\boldsymbol{x}) - f_{\boldsymbol{y}}(\boldsymbol{x})\right| \leq \varepsilon$ with success probability at least $1 - \delta$.

It is easy to see that this problem corresponds to a simple Monte Carlo mean estimation. Indeed, we can write a decomposition of $O(\boldsymbol{y})$ in its eigenbasis as:

$$O(\boldsymbol{y}) = \sum_i \lambda_i(\boldsymbol{y}) \, |\phi_i\rangle\langle\phi_i| \tag{6.23}$$

where $\lambda_i(\boldsymbol{y})$ is a real eigenvalue (since $O(\boldsymbol{y})$ is Hermitian) associated to the eigenstate $|\phi_i\rangle\langle\phi_i|$ (these eigenstates can in general also depend on $\boldsymbol{y}$,

but we do not write this dependence explicitly for ease of notation). We can also write a decomposition of $\rho(\boldsymbol{x})$ in this same basis as :

$$\rho(\boldsymbol{x}) = \sum_{i,j} \rho_{i,j}(\boldsymbol{x}) \left|\phi_i\right\rangle\!\left\langle\phi_j\right| \tag{6.24}$$

such that $\mathrm{Tr}[\rho(\boldsymbol{x})] = \sum_i \rho_{i,i}(\boldsymbol{x}) = 1$ by the unit-trace property of $\rho(\boldsymbol{x})$, and $\left\langle\phi_i\right| \rho(\boldsymbol{x}) \left|\phi_i\right\rangle = \rho_{i,i}(\boldsymbol{x}) \geq 0$ from its positive semi-definiteness. From these two properties, we deduce that $\{\rho_{i,i}(\boldsymbol{x})\}_i$ defines a probability distribution over the eigenstates $\{\left|\phi_i\right\rangle\!\left\langle\phi_i\right|\}_i$. Therefore, we can see that:

$$\mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})] = \mathrm{Tr}\left[\sum_{i,j,k} \rho_{i,j}(\boldsymbol{x})\lambda_k(\boldsymbol{y}) \left|\phi_i\right\rangle\!\left\langle\phi_j\middle|\phi_k\right\rangle\!\left\langle\phi_k\right|\right] \tag{6.25}$$

$$= \mathrm{Tr}\left[\sum_{i,j} \rho_{i,j}(\boldsymbol{x})\lambda_j(\boldsymbol{y}) \left|\phi_i\right\rangle\!\left\langle\phi_j\right|\right] \tag{6.26}$$

$$= \sum_i \rho_{i,i}(\boldsymbol{x})\lambda_i(\boldsymbol{y}) \tag{6.27}$$

simply corresponds to the expectation value of the random variable $i \mapsto \lambda_i(\boldsymbol{y})$ under the probability distribution $\{\rho_{i,i}(\boldsymbol{x})\}_i$, i.e., the probability distribution obtained by measuring $\rho(\boldsymbol{x})$ in the eigenbasis of $O(\boldsymbol{y})$.

Therefore, we can use known results from (classical) Monte Carlo estimation to bound the sample complexity of evaluating this mean value, and therefore the quantum model. With the assumption that $\rho(\boldsymbol{x})$ is given as a black-box and that it can generate arbitrary quantum states (and therefore arbitrary distributions $\{\rho_{i,i}(\boldsymbol{x})\}_i$), the only property of the random variable $i \mapsto \lambda_i(\boldsymbol{y})$ we can use to bound the sample complexity is its bounded domain. Indeed, without additional assumptions of the distribution, we have a tight sample complexity bound of

$$\Theta\left(\frac{B^2 \log\left(\delta^{-1}\right)}{\varepsilon^2}\right) \tag{6.28}$$

samples in order to estimate the mean of a random variable taking values in $[-B, B]$, to precision ε and with probability of success $1 - \delta$ [134, 135]. In the case of a quantum model, the random $i \mapsto \lambda_i(\boldsymbol{y})$ takes values in $[-\|O(\boldsymbol{y})\|_\infty, \|O(\boldsymbol{y})\|_\infty]$, where $\|O(\boldsymbol{y})\|_\infty$ is the spectral norm of the observable $O(\boldsymbol{y})$. Therefore, the sample complexity of estimating a

quantum model $f_{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})]$ is in

$$\Theta\left(\frac{\|O(\boldsymbol{y})\|_\infty^2 \log(\delta^{-1})}{\varepsilon^2}\right), \tag{6.29}$$

in the absence of any constraint (or information) on the quantum states $\rho(\boldsymbol{x})$.

6.B.2 Generalization performance

In this section, we study the generalization performance of flipped models. Our result can be stated informally in the following lemma:

Lemma 6.B.1 (Generalization bounds (informal))
Consider a flipped model $f_{\boldsymbol{\theta}}$ that acts on n qubits and has a bounded observable norm $\|O\|_\infty$. If this model achieves a small training error $|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \eta$ for all $\boldsymbol{x}$ in a dataset of size M, then it also has a small expected error $|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq 2\eta$ with probability $1-\varepsilon$ over the data distribution, provided that the size of the dataset scales as $M \geq \widetilde{\Omega}\left(\frac{n\|O\|_\infty^2}{\varepsilon\eta^2}\right)$.

To prove this result, we take an approach very similar to that of Aaronson [136], where we lower bound the number of qubits n and spectral norm $\|O\|_\infty$ needed by a flipped model to encode arbitrary k-bit strings, in a way that can be recovered efficiently via repeated measurements. These bounds naturally allow us to upper bound the fat-shattering dimension of flipped models, a complexity measure that is widely used in generalization bounds [137].

Encoding bit-strings in flipped models

Theorem 6.B.1
Let k and n be positive integers with $k > n$. For all k-bit strings $\boldsymbol{y} = y_1 \ldots y_k$, let $\rho(\boldsymbol{y})$ be an n-qubit mixed state that "encodes" $\boldsymbol{y}$, meaning each bitstring $\boldsymbol{y}$ is associated to an arbitrary n-qubit quantum state $\rho(\boldsymbol{y})$. Suppose there exist Hermitian observables $O_1, \ldots, O_k$ with spectral norms $\|O_i\|_\infty$ such that we call $\|O\|_\infty = \max_i \|O_i\|_\infty$, as well as real numbers $\alpha_1, \ldots, \alpha_k$, such that, for all $\boldsymbol{y} \in \{0,1\}^k$ and $i \in \{1, \ldots, k\}$,

(i) if $y_i = 0$, then $\mathrm{Tr}[\rho(\boldsymbol{y})O_i] \leq \alpha_i - \gamma$, and

(ii) if $y_i = 1$, then $\mathrm{Tr}[\rho(\boldsymbol{y})O_i] \geq \alpha_i + \gamma$.

Then $n\|O\|_\infty^2/\gamma^2 \in \Omega(k)$.

Proof. We take a similar approach to the proof of Aaronson [136], in which we show that a combination of an encoding $\rho(\boldsymbol{y})$ and observables $O_1, \ldots, O_k$ that satisfies guarantees (i) and (ii) would need $n\|O\|_\infty^2/\gamma^2$ to scale linearly in the length k of the bit-strings it encodes in order not to contradict with Holevo's bound.

Suppose by contradiction that such an encoding scheme exists with $n\|O\|_\infty^2/\gamma^2 \in o(k)$. We first adapt to the setting of Aaronson by constructing two-outcome POVMs $\{E_i, I - E_i\}$ out of the observables O_i. That is, we take the general Hermitian matrices O_i with eigenvalues in $[\lambda_{\min}, \lambda_{\max}] \subset [-\|O\|_\infty, \|O\|_\infty]$, and transform them into Hermitian matrices E_i with eigenvalues in $[0, 1]$, such that the POVM $\{E_i, I - E_i\}$ accepts ρ (i.e., outputs 1) with probability $\mathrm{Tr}(\rho E_i)$, and rejects ρ (i.e., outputs 0) with probability $1 - \mathrm{Tr}(\rho E_i)$. Specifically, we define

$$E_i = \frac{O_i + |\lambda_{\min}| I}{|\lambda_{\min}| + |\lambda_{\max}|}. \tag{6.30}$$

Conditions (i) and (ii) then translate to:

$$\begin{cases} (i') \text{ if } y_i = 0, \text{ then } \mathrm{Tr}[\rho(\boldsymbol{y}) E_i] \leq \frac{\alpha_i + |\lambda_{\min}|}{|\lambda_{\min}| + |\lambda_{\max}|} - \frac{\gamma}{|\lambda_{\min}| + |\lambda_{\max}|} \\ (ii') \text{ if } y_i = 1, \text{ then } \mathrm{Tr}[\rho(\boldsymbol{y}) E_i] \geq \frac{\alpha_i + |\lambda_{\min}|}{|\lambda_{\min}| + |\lambda_{\max}|} + \frac{\gamma}{|\lambda_{\min}| + |\lambda_{\max}|} \end{cases} \tag{6.31}$$

From here, by noting that $|\lambda_{\min}| + |\lambda_{\max}| \leq 2\|O\|_\infty$, we can directly apply Theorem 2.6 of Aaronson [136] and get our result, but we detail the reasoning further for clarity.

We first need to amplify the probability that we correctly identify whether $y_i = 0$ or 1 from measuring copies of $\rho(\boldsymbol{y})$, since the probabilities obtained from E_i can be arbitrarily small. Consider an amplified scheme, where each bit-string $\boldsymbol{y} \in \{0, 1\}^k$ is encoded by the tensor product $\rho(\boldsymbol{y})^{\otimes \ell}$, for some $\ell > 1$ to be defined later. For all $i \in \{1, \ldots, k\}$, let $\{E_i^*, I - E_i^*\}$ be the amplified POVM that applies $\{E_i, I - E_i\}$ to each of the ℓ copies of $\rho(\boldsymbol{y})$ and accepts if and only if at least $\widetilde{\alpha}_i \ell = \frac{\alpha_i + |\lambda_{\min}|}{|\lambda_{\min}| + |\lambda_{\max}|} \ell$ of these POVMs do. For all $j \in \{1, \ldots, \ell\}$, call $X_i^{(j)}$ the random variable that takes the value 1 if $\{E_i, I - E_i\}$ accepts the j-th copy of $\rho(\boldsymbol{y})$ (i.e., with probability $p_i = \mathrm{Tr}(\rho(\boldsymbol{y}) E_i)$), and value 0 otherwise.

Consider the case where $y_i = 0$. We have $\mathrm{Tr}[O_i \rho(y)] \leq \alpha_i - \gamma$, which implies that $\widetilde{\alpha}_i \geq p_i + \frac{\gamma}{|\lambda_{\min}| + |\lambda_{\max}|}$. Therefore, the probability that at

least $\widetilde{\alpha}_i \ell$ of the POVMs accept is then:

$$P(\overline{X}_i \geq \widetilde{\alpha}_i) \leq P\left(\overline{X}_i \geq p_i + \frac{\gamma}{|\lambda_{\min}| + |\lambda_{\max}|}\right) \tag{6.32}$$

for $\overline{X}_i = \frac{1}{\ell}\sum_{j=1}^{\ell} X_i^{(j)}$. From the Chernoff bound, we hence get:

$$P(\overline{X}_i \geq \widetilde{\alpha}_i) \leq e^{-2\left(\frac{\gamma}{|\lambda_{\min}|+|\lambda_{\max}|}\right)^2 l}. \tag{6.33}$$

To guarantee an acceptance probability $\mathrm{Tr}(\rho(\boldsymbol{y})^{\otimes \ell} E_i^*) \leq 1/3$, it is then sufficient to take $\ell = \left\lceil \frac{2\log(3)\|O\|_\infty^2}{\gamma^2} \right\rceil$. A similar analysis holds for the case $y_i = 1$.

From here, the result we use that derives from Holevo's bound is Theorem 5.1 of Ambainis *et al.* [138]. It states that in order for the POVMs $\{E_i^*, I - E_i^*\}$ to correctly identify whether $y_i = 0$ or 1 with probability of failure less than $1/3$, we need a number of qubits $n\ell \geq (1 - H(1/3))k$, where H is the binary entropy function. This implies that $n\|O\|_\infty^2/\gamma^2 \geq \frac{2(1-H(1/3))}{\log(3)} k \in \Omega(k)$. □

Generalization bounds of flipped models

The conditions (i) and (ii) of Theorem 6.B.1 are very similar to that of a fat-shattering dimension of a concept class.

Definition 6.B.1
Let $\mathcal{X}$ be a data space, let $\mathcal{C}$ be a class of functions from $\mathcal{X}$ to $\mathbb{R}$, and let $\gamma > 0$. The fat-shattering dimension of the concept class $\mathcal{C}$ at width γ, denoted $\mathrm{fat}_{\mathcal{C}}(\gamma)$, is defined as the size k of the largest set of points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}\}$ for which there exist real numbers $\alpha_1, \ldots, \alpha_k$ such that for all $\boldsymbol{y} \in \{0,1\}^k$, there exists a $f \in \mathcal{C}$ that satisfies, for all $i \in \{1, \ldots, k\}$,

(i) if $y_i = 0$, then $f(\boldsymbol{x}) \leq \alpha_i - \gamma$, and

(ii) if $y_i = 1$, then $f(\boldsymbol{x}) \geq \alpha_i + \gamma$.

By comparing this definition with the statement of Theorem 6.B.1, we can show:

Corollary 6.B.2
Consider a flipped model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ defined on a data space $\mathcal{X}$, using n-qubit quantum states and observables with spectral norm $\|O\|_\infty =$

$\sup_{\boldsymbol{x}\in\mathcal{X}} \|O(\boldsymbol{x})\|_\infty$. *Call* $\mathcal{C}_{n,O} = \{f_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}}$ *the concept (or hypothesis) class associated to this model. Then, for all* $\gamma > 0$, *we have* $\mathrm{fat}_{\mathcal{C}_{n,O}}(\gamma) \in \mathcal{O}(n\|O\|_\infty^2/\gamma^2)$.

Proof. We note that since, for all $\boldsymbol{x} \in \mathcal{X}$, $O(\boldsymbol{x})$ lives in a manifold of all observables with spectral norm $\|O\|_\infty$, then the fat-shattering dimension of $\mathcal{C}_{n,O}$ is upper bounded by that of the concept class $\widetilde{\mathcal{C}}_{n,O} = \{O' \mapsto \mathrm{Tr}[\rho(\boldsymbol{\theta})O']\}_{\boldsymbol{\theta}}$ defined on the input space of observables O' that satisfy $\|O'\|_\infty \leq \|O\|_\infty$. Theorem 6.B.1 immediately yields an upper bound for $\mathrm{fat}_{\widetilde{\mathcal{C}}_{n,O}}(\gamma)$, when we identify $\rho(\boldsymbol{\theta})$ in this corollary to $\rho(\boldsymbol{y})$ in the Theorem, and observe that the number of observables $O_1, \ldots, O_k$ that can be γ-shattered (i.e., conditions (i) and (ii) for all labelings) must satisfy $k \in \mathcal{O}(n\|O\|_\infty^2/\gamma^2)$. □

To obtain generalization bounds on the performance of flipped models, we combine this bound on their fat-shattering dimension with standard results from learning theory, e.g.:

Theorem 6.B.3 (Anthony and Barlett [137])
Let $\mathcal{X}$ *be a data space, let* $\mathcal{C}$ *be a class of functions from* $\mathcal{X}$ *to* $\mathbb{R}$, *and let* $\mathcal{D}$ *be a probability measure over* $\mathcal{X}$. *Fix an element* $f \in \mathbb{R}^{\mathcal{X}}$, *as well as error parameters* $\varepsilon, \eta, \gamma, \delta > 0$ *with* $\gamma > \eta$. *Suppose that we draw* m *samples* $X = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$ *from* $\mathcal{X}$ *according to* $\mathcal{D}$, *and choose any hypothesis* $h \in \mathcal{C}$ *such that* $|h(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \eta$ *for all* $\boldsymbol{x} \in X$. *Then, there exists a positive constant* K *such that, provided*

$$m \geq \frac{K}{\varepsilon}\left(\mathrm{fat}_{\mathcal{C}}\left(\frac{\gamma-\eta}{8}\right)\log^2\left(\frac{\mathrm{fat}_{\mathcal{C}}\left(\frac{\gamma-\eta}{8}\right)}{(\gamma-\eta)\varepsilon}\right) + \log\left(\frac{1}{\delta}\right)\right),$$

with probability $1-\delta$,

$$\Pr_{\boldsymbol{x}\in\mathcal{D}}[|h(\boldsymbol{x}) - f(\boldsymbol{x})| > \gamma] \leq \varepsilon.$$

From Corollary 6.B.2, we have that

$$m \in \widetilde{\Omega}\left(\frac{1}{\varepsilon}\left(\frac{n\|O\|_\infty^2}{(\gamma-\eta)^2} + \log\left(\frac{1}{\delta}\right)\right)\right)$$

samples suffice to get these generalization guarantees. This proves Lemma 6.B.1.

6.B.3 Flipping bounds

In this section, we study mappings between conventional and flipped models (most importantly from conventional to flipped, but our flipping bounds can be used either way). We find that the important quantity that governs the trade-off in resources between these models is the observable trace norm $\|O\|_1 = \mathrm{Tr}\left[\sqrt{O^2}\right]$ of the model to be mapped. Since all observables $O(\boldsymbol{y})$ (where $\boldsymbol{y}$ is either a data vector or a parameters vector, depending on the model) have to be turned into unit-trace density matrices, their eigenvalues need (i) either to be normalized when preserving the number of qubits of the model, or (ii) be encoded in more qubits than in the original model (e.g., by using a binary encoding of all the eigenvalues of $O(\boldsymbol{y})$). Each of these options has its disadvantages: (i) normalizing eigenvalues introduces an overhead in the spectral norm $\|O'\|_\infty$ of the observable of the resulting model, which results in an overhead in the number of measurements needed to evaluate this model to the same precision as the original one. As for (ii), we show that the number of qubits of the new model would need to scale quadratically with $\|O\|_1$ in general, which is commonly an exponential quantity in the number of qubits of the original model (e.g., when O is a Pauli). We show the following lemma:

Lemma 6.B.2 (Flipping bounds (informal))
Any conventional linear model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$ acting on n qubits and with a bounded observable trace norm $\|O\|_1 \leq d$ admits an equivalent flipped model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})]$ acting on $m = n+1$ qubits and with observable spectral norm $\|O'\|_\infty = d$. The bound on the spectral norm is essentially tight in the regime where $n, m \in \mathcal{O}(\log(d))$, i.e., in this case we have $\|O'\|_\infty \geq \widetilde{\Omega}(d)$.

Upper bounds

Theorem 6.B.4
Given a specification of a conventional quantum model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$ acting on n qubits, with a known (upper bound on the) trace norm $\|O\|_1$ of its observable, one can construct an equivalent flipped model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})]$ acting on $n+1$ qubits such that $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, $\forall \boldsymbol{x}, \boldsymbol{\theta}$ and $\|O'\|_\infty = \|O\|_1$.

Proof. From Def. 6.A.1, we assume that, in the definition of the conventional model, $\rho(\boldsymbol{x})$ is obtained by applying a unitary $U(\boldsymbol{x})$ on a known quantum state ρ_0 that is diagonal in the computational basis (i.e., is a mixture of computational basis states). As for $O(\boldsymbol{\theta})$, we assume that it is

specified by a unitary $V(\boldsymbol{\theta})$ and a weighted sum of d Hermitian operators O_i, such that $O(\boldsymbol{\theta}) = \sum_{i=1}^{d} w_i V^\dagger(\boldsymbol{\theta}) O_i V(\boldsymbol{\theta})$ and for each $i \in \{1, \dots, d\}$ we know how to decompose O_i as $O_i = \sum_{j=0}^{2^n-1} \lambda_{i,j} W_i^\dagger |j\rangle\langle j| W_i$, for some known $\lambda_{i,j}$'s and W_i's. Note that, as opposed to the parameters that specify $V(\boldsymbol{\theta})$, the weights w_i influence the trace norm $\|O(\boldsymbol{\theta})\|_1$. Therefore we need to pay attention to the fact that $\|O(\boldsymbol{\theta})\|_1$ is only upper bounded by $\|O\|_1 = \sup_{\boldsymbol{\theta}} \|O(\boldsymbol{\theta})\|_1$, and that these two quantities are not always equal.

Out of the observables $O(\boldsymbol{\theta})$, we need to prepare quantum states $\rho'(\boldsymbol{\theta})$ such that $\mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})] = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$. The only difficulty is that quantum states are positive semi-definite and have unit trace while Hermitian observables generally do not fulfill any of these two conditions. To get around these constraints, we simply decompose the observables $O(\boldsymbol{\theta})$ into positive and negative components, that we both normalize. More precisely, call $O_+(\boldsymbol{\theta})$ ($O_-(\boldsymbol{\theta})$) the positive (negative) part of $O(\boldsymbol{\theta}) = O_+(\boldsymbol{\theta}) - O_-(\boldsymbol{\theta})$. We define:

$$\begin{cases} \rho'_+(\boldsymbol{\theta}) = O_+(\boldsymbol{\theta})/\|O_+(\boldsymbol{\theta})\|_1 \\ \rho'_-(\boldsymbol{\theta}) = O_-(\boldsymbol{\theta})/\|O_-(\boldsymbol{\theta})\|_1 \end{cases} \quad \text{and} \quad \begin{cases} p_+ = \|O_+(\boldsymbol{\theta})\|_1/\|O(\boldsymbol{\theta})\|_1 \\ p_- = \|O_-(\boldsymbol{\theta})\|_1/\|O(\boldsymbol{\theta})\|_1 \end{cases} \tag{6.34}$$

such that

$$\rho'(\boldsymbol{\theta}) = p_+ |0\rangle\langle 0| \otimes \rho'_+(\boldsymbol{\theta}) + p_- |1\rangle\langle 1| \otimes \rho'_-(\boldsymbol{\theta}) \tag{6.35}$$

is a valid quantum state (positive semi-definite and unit trace). We can then take

$$O'(\boldsymbol{x}) = \|O(\boldsymbol{\theta})\|_1 (|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes \rho(\boldsymbol{x}) \tag{6.36}$$

which, as one can easily verify, leads to $\mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})] = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$. However, this still does not give us a proper flipped model as the renormalization factor $\|O(\boldsymbol{\theta})\|_1$ of $O'(\boldsymbol{x})$ can depend on the parameters $\boldsymbol{\theta}$ (and more precisely the weights w_i, see remark above). We would like to use here the upper bound $\|O\|_1$. To do so, we can simply (re-)define:

$$\begin{cases} p_+ = \|O_+(\boldsymbol{\theta})\|_1/\|O\|_1 \\ p_- = \|O_-(\boldsymbol{\theta})\|_1/\|O\|_1 \end{cases} \quad \text{and} \quad p_0 = \frac{\|O\|_1 - \|O_+(\boldsymbol{\theta})\|_1 - \|O_-(\boldsymbol{\theta})\|_1}{\|O\|_1} \tag{6.37}$$

and also

$$\begin{cases} \rho'(\boldsymbol{\theta}) = p_+ |0\rangle\langle 0| \otimes \rho'_+(\boldsymbol{\theta}) + p_- |1\rangle\langle 1| \otimes \rho'_-(\boldsymbol{\theta}) + p_0 I/2^{n+1} \\ O'(\boldsymbol{x}) = \|O\|_1 (|0\rangle\langle 0| - |1\rangle\langle 1|) \otimes \rho(\boldsymbol{x}) \end{cases} \tag{6.38}$$

such that we still have $\mathrm{Tr}[\rho'(\boldsymbol{\theta})O'(\boldsymbol{x})] = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$ but where we have now defined a proper flipped model. □

Going a bit further, we also propose an algorithm to evaluate the flipped model we constructed in our proof (see Algorithm 1). Given that we only assume to know the eigenvalue decomposition of the single O_i's, and not that of the full observable $O(\boldsymbol{\theta})$, we do not decompose $O(\boldsymbol{\theta})$ directly into its positive and negative components, but rather the O_i's. For this, we define:

$$\begin{cases} O_{i,+} = \sum_{j,\lambda_{i,j}\geq 0} \lambda_{i,j} W_i^\dagger |j\rangle\langle j| W_i \\ O_{i,-} = \sum_{j,\lambda_{i,j}\leq 0} |\lambda_{i,j}| W_i^\dagger |j\rangle\langle j| W_i \end{cases} \tag{6.39}$$

We then recover $\rho'(\boldsymbol{\theta})$ via importance sampling of the indices i, j and $\pm$ and the implementation of the pure state $V(\boldsymbol{\theta})W_i|j\rangle$ (see Algorithm 1)[4]. Naturally, one could alternatively design a full unitary implementation of $\rho'(\boldsymbol{\theta})$ using auxiliary qubits to prepare coherent encodings of the probability distributions appearing in Algorithm 1, along with controlled operations between these auxiliary qubits and the working register, but this would require more qubits and a more complicated quantum implementation.

Lower bounds

Theorem 6.B.5
For d an arbitrary positive integer, there exists a conventional model $\mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})]$, acting on $n \in \mathcal{O}(\log(d))$ qubits, that satisfies $\|O(\boldsymbol{y})\|_1 = d$ for all $\boldsymbol{y} \in \mathcal{Y}$, such that for any flipped model $\mathrm{Tr}[\rho'(\boldsymbol{y})O'(\boldsymbol{x})]$ acting on m qubits with $\|O'\|_\infty = \sup_{\boldsymbol{x}\in\mathcal{X}} \|O'(\boldsymbol{x})\|_\infty$, and any $\varepsilon \geq 0$, if the model satisfies

$$|\mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})] - \mathrm{Tr}[\rho'(\boldsymbol{y})O'(\boldsymbol{x})]| \leq \varepsilon \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \times \mathcal{Y} \tag{6.40}$$

then, it must also satisfy

$$m\|O'\|_\infty^2 \in \Omega(d^2(1/2-\varepsilon)^2).$$

Proof. The core of the proof is to show that a conventional model $\mathrm{Tr}[\rho(i)O(\boldsymbol{y})]$ with trace norm $\|O(\boldsymbol{y})\|_1 = d$ and acting on $n = \lceil \log_2(N+1) \rceil$ qubits, for $N = \lfloor d^2/4 \rfloor$, can represent the function $i \mapsto y_i$ for $1 \leq i \leq N$, for all

[4] In Algorithm 1, $\sigma(\boldsymbol{\theta})$ corresponds to either $\rho'_+(\boldsymbol{\theta})$ or $\rho'_-(\boldsymbol{\theta})$, depending on the sampled $b \in \{+,-\}$.

$\boldsymbol{y} \in \{0,1\}^N$. For this, we take, for all $1 \leq i \leq N$:

$$\rho(i) = \frac{1}{2}(|0\rangle + |i\rangle)(\langle 0| + \langle i|) \quad \text{and} \quad O(\boldsymbol{y}) = \sum_{i'=1}^{N+1} O_{i'}(\boldsymbol{y}) \tag{6.41}$$

for

$$O_i(\boldsymbol{y}) = \begin{cases} y_i(|0\rangle\langle i| + |i\rangle\langle 0|) & \text{if } 1 \leq i \leq N \\ (d - \sqrt{|\boldsymbol{y}|})\,|N+1\rangle\langle N+1| & \text{if } i = N+1 \end{cases} \tag{6.42}$$

where $|\boldsymbol{y}| = \sum_{i=1}^N y_i$ is the Hamming weight of $\boldsymbol{y}$. By construction, we have that the upper-left $N \times N$ block of $O(\boldsymbol{y})$ satisfies $\left\|O_{(N\times N)}(\boldsymbol{y})\right\|_1 = \sqrt{|\boldsymbol{y}|} \leq d$ (it corresponds to the adjacency matrix of a star graph of degree $D = |\boldsymbol{y}| \leq N$, which has trace norm $2\sqrt{D}$ [139]), such that $\|O(\boldsymbol{y})\|_1 = d$ for all $\boldsymbol{y} \in \{-1,1\}^N$. Also, it is easy to check that $\mathrm{Tr}[\rho(i)O(\boldsymbol{y})] = y_i$ for all $1 \leq i \leq N$.
We take this conventional model to be our target model. Satisfying the condition of Eq. (6.40) is then equivalent to satisfying:

$$|\mathrm{Tr}[\rho'(\boldsymbol{y})O'(i)] - y_i| \leq \varepsilon \quad \forall\, i, \boldsymbol{y} \in \{1,\ldots,N\} \times \{0,1\}^N.$$

Now note that this condition is stronger than that of Theorem 6.B.1 for $\gamma = 1/2 - \varepsilon$ and $\alpha_{i,j} = 1/2\ \forall i,j$. Therefore, in order not to contradict with this theorem, we must have $m\|O'\|_\infty^2 \in \Omega(N(1/2-\varepsilon)^2) = \Omega(d^2(1/2-\varepsilon)^2)$. □

Lemma 6.B.2 is obtained from Theorem 6.B.4 as stated above and Theorem 6.B.5 for the case $m \in \mathcal{O}(\log d)$, such that the statement $m\|O'\|_\infty^2 \in \Omega(d^2(1/2-\varepsilon)^2)$ becomes $\|O'\|_\infty \in \Omega\left(\frac{d}{\sqrt{\log(d)}}(1/2-\varepsilon)\right)$.

Circumventing lower bounds

Note that our flipping bounds are essentially tight only in the regime where the number of qubits used by the original and resulting model n, m are both in $\mathcal{O}(\mathrm{polylog}(\|O\|_1))$. This is a relevant regime, as it includes notably the case of Pauli observables (be they local or non-local) or linear combinations thereof. However, outside this regime our bounds can be circumvented.

Also note that an easy way of circumventing our lower bounds, even in the regime where $n, m \in \mathcal{O}(\mathrm{polylog}(\|O\|_1))$, is by imposing the constraint

that $O(\boldsymbol{y})$ is parametrized by $|\boldsymbol{y}| = \mathcal{O}(\text{poly}(n))$ parameters[5]. Indeed, in this case, one can simply use a similar construction to that in Ref. [114] (Fig. 3) where one would encode $\boldsymbol{y}$ (e.g., in binary form) in auxiliary qubits as $|\widetilde{\boldsymbol{y}}\rangle$ and use controlled operations that are independent of $\boldsymbol{y}$ to simulate the action of gates parametrized by $\boldsymbol{y}$. One can then note that by taking $\rho'(\boldsymbol{\theta}) = \rho_0 \otimes |\widetilde{\boldsymbol{y}}\rangle\langle\widetilde{\boldsymbol{y}}|$ and the rest of the resulting circuit to define $O'(\boldsymbol{x})$, one ends up with a flipped model that acts on $\mathcal{O}(\text{poly}(n))$ qubits and satisfies $\|O'\|_\infty = \|O\|_\infty$. For O defined by a Pauli observable for instance, we have $\|O\|_\infty = 1$ and $\|O\|_1 = 2^n$. Such a construction therefore does not suffer from an exploding spectral norm $\|O'\|_\infty$. However, it also does not lead to shadowfiable models in general as the parametrized states $\rho'(\boldsymbol{\theta})$ play a trivial role and the observables $O'(\boldsymbol{x})$ hide all of the quantum computation.

6.C Quantum advantage using shadow models

6.C.1 Discrete cube root learning task

In this section we rigorously define the discrete cube root learning task introduced in the main text and detail the proof of its classical hardness. More precisely, we start by introducing the discrete cube root problem and state formally its classical hardness assumption (the discrete cube root assumption). Then we construct a learning task that is classically hard based on this assumption.

The discrete cube root problem

A definition of the discrete cube root problem can be found in Ref. [125]. For convenience, we restate it in this appendix.[6] Consider two large prime numbers p and q of the form $3k+2$, $3k'+2$, for distinct k, k', and which can be represented by approximately the same number of bits. Let $N = pq$ be the product of these primes, which we assume to be an n-bit integer, and let $\mathbb{Z}_N = \{0, \ldots, N-1\}$.

[5] In our proof of Theorem 6.B.5, we use $|\boldsymbol{y}| \in \Omega(\|O\|_1^2)$, which is in $\Omega(\exp(n))$ for $n, m \in \mathcal{O}(\log(\|O\|_1))$, and subexponential in n for $n, m \in \mathcal{O}(\text{polylog}(\|O\|_1))$

[6] Note that, as opposed to the exposition of Ref. [125], we consider the domain $\mathbb{Z}_N = \{0, \ldots, N-1\}$ instead of $\{i \mid 0 < i < N,\ \gcd(i, N) = 1\}$ for the functions we define next. This allows us to apply more easily the result of Ref. [127] and construct a learning task with a stronger form of classical hardness.

We consider the "discrete cube" function $f_N(y) : \mathbb{Z}_N \to \mathbb{Z}_N$ defined as $f_N(y) = y^3 \bmod N$, as well as its inverse, which we denote as $g_N(x) = f_N^{-1}(x) = \sqrt[3]{x} \bmod N$ (see Fig. 6.3). As we explain in the following, f_N is particularly interesting because it is believed to be a *one-way function*: $f_N(y)$ can be computed efficiently classically using modular exponentiation, while $g_N(x)$ is believed hard to compute classically, with only knowledge of N and x (and not the factors p, q of N). But first, let us show that the inverse function g_N is properly defined.

Lemma 6.C.1
For p, q two distinct prime numbers of the form $3k+2$, $3k'+2$ and $N = pq$, the function $f_N(y) = y^3 \mod N$ is a bijection on $\mathbb{Z}_N$.

Proof. To show that f_N is a bijection on $\mathbb{Z}_N$, it is sufficient to show that it is injective, since it maps $\mathbb{Z}_N$ to itself.
Consider $y, z \in \mathbb{Z}_N$ such that $f_N(y) = f_N(z)$, i.e., $y^3 \bmod N = z^3 \bmod N$. We then have $y^3 - z^3 \equiv 0 \bmod N$, which means that there exists an $l \in \mathbb{N}$ such that $y^3 - z^3 = lN = lpq$. Therefore, we know that $y^3 - z^3$ is divisible by both p and q, which implies $y^3 \equiv z^3 \bmod p$ and $y^3 \equiv z^3 \bmod q$. From here, we apply a similar reasoning for p and q, that we detail only for p.
Given that we assumed $p = 3k + 2$, we know that $\gcd(p - 1, 3) = 1$. Therefore, Euclid's algorithm assures that there exist $d_p, d'_p \geq 1$ such that $3d_p = (p - 1)d'_p + 1$. Then notice that:

$$y^{3d_p} \equiv y^{(p-1)d'_p+1} \equiv y \bmod p \tag{6.43}$$

where the last congruence follows from applying Fermat's little theorem ($y^p \equiv y \bmod p$ for all $y \in \mathbb{N}$) to show by induction on $m \geq 0$ that $y^{(p-1)m+1} \equiv y \bmod p$. Similarly, $z^{3d_p} \equiv z \bmod p$, and therefore by raising to the power d_p both terms in $y^3 \equiv z^3 \bmod p$, we get $y \equiv z \bmod p$.
From the same reasoning, we get $y \equiv z \bmod q$, which implies that $x - y$ is divisible by both p and q. Since they are distinct primes, then $x - y$ is also divisible by $pq = N$, which means that $x \equiv y \bmod N$. This shows that f_N is injective, and therefore bijective. □

Now that we have shown that the discrete cube root function is properly defined on $\mathbb{Z}_N$, let us define the discrete cube root problem:

Definition 6.C.1 (Discrete cube root problem)
Let p and q be two distinct primes of the form $3k + 2, 3k' + 2$ represented by approximately the same number of bits, and such that $N = pq$ is an n-bit integer. Given as input both N and $x \in \mathbb{Z}_N$, output $y \in \mathbb{Z}_N$ such that $y^3 = x \bmod N$.

The assumption that the "discrete cube" function f_N is a one-way function is formalized by the so-called discrete cube root assumption, which is a special case of the RSA assumption for the exponent $e = 3$.

Definition 6.C.2 (Discrete cube root assumption [125])
For any polynomial $P(.)$, there does not exist a classical algorithm $\mathcal{A}$, that runs in time $P(n)$ and that, on input N and x (where N is the n-bit product of two random primes of the form $3k+2$ and x is chosen randomly from $\mathbb{Z}_N$), outputs $y \in \mathbb{Z}_N$ such that with probability $1/P(n)$ satisfies $y^3 = x \mod N$. The probability of success is taken over the random draws of the two primes p, q and the input $x \in \mathbb{Z}_N$ and any internal randomisation of $\mathcal{A}$.

While other one-way functions like the discrete exponential (along with its inverse, the discrete logarithm) also have similar classical hardness assumptions, the discrete cube (root) function is additionally known to be a *trap-door function*. That is, there exists a key $d \in \mathbb{Z}_N$ such that $g_N(x)$ can be alternatively computed via modular exponentiation as $g_N(x) = x^d \mod N$. This key d can be efficiently computed from the prime factors p and q of N. We show this using a similar reasoning to that around Equation 6.43. Call $\phi(N) = (p-1)(q-1)$. From $p = 3k+2$ and $q = 3k'+2$, we have $\gcd(\phi(N), 3) = 1$. Therefore, Euclid's algorithm assures that there exists $d, d' \geq 1$ such that $3d = \phi(N)d' + 1 = (p-1)(q-1)d' + 1$. We want to show that, for all $y \in \mathbb{Z}_N$,

$$y^{3d} \equiv y \mod N. \tag{6.44}$$

To do this, we show that $y^{3d} - y$ is divisible by both p and q. In the case of p, we have:

$$y^{3d} \equiv y^{(p-1)(q-1)d'+1} \equiv y \mod p. \tag{6.45}$$

The last equality follows again from Fermat's little theorem (see Equation 6.43). Similarly, $y^{3d} \equiv y \mod q$, which implies that $y^{3d} - y$ is divisible by p and q, and that $y^{3d} \equiv y \mod N$. Therefore d is a valid key for computing $g_N(y)$ for all $y \in \mathbb{Z}_N$. When knowing the factors p and q of N, one can compute $\phi(N)$ and use Euclid's algorithm to find d such that $3d = 1 \mod \phi(N)$. However, factoring a large N is believed to be computationally intractable classically, which justifies the discrete cube root assumption (Def. 6.C.2).

The learning task

Based on the discrete cube root assumption, we can construct a learning task that is not efficiently PAC learnable classically. In order to define the concept class of this learning task, we first consider the class of functions:

$$\mathcal{F}_n = \{g_N : \mathbb{Z}_N \to \mathbb{Z}_N \mid g_N(x) = \sqrt[3]{x} \bmod N, \\ \text{for } N \text{ an } n\text{-bit integer that satisfies the DCR conditions}\}$$

defined for any integer n. We define the concept class:

$$\mathcal{C}_n = \{g_{N,s} : \mathbb{Z}_N \to \{0,1\} \mid g_{N,s}(x) = \begin{cases} 1, & \text{if } g_N(x) \in [s, s + \frac{N-1}{2}], \\ 0, & \text{otherwise.} \end{cases} \tag{6.46}$$

$$\text{, for } g_N \in \mathcal{F}_n,\ s \in \mathbb{Z}_N\} \tag{6.47}$$

for any integer n. In our proofs of classical hardness and quantum learnability, we also need that the integer N corresponding to the target function should be specified with the training data. One way of doing so is to append it to all inputs $x \in \mathbb{Z}_N$ as (x, N) and redefine the concept class accordingly. To ease notation, we assume this transformation to be done implicitly in the following.

Because the discrete cube function is a one-way function and is bijective on $\mathbb{Z}_N$, it is easy to classically generate training data for any concept $g_{N,s} \in \mathcal{C}_n$, under the uniform distribution over $\mathbb{Z}_N$. Indeed, one can simply uniformly sample $y \in \mathbb{Z}_N$, compute its corresponding $x = y^3 \bmod N$ via modular exponentiation and keep from y only the label indicating whether $y \in [s, s + \frac{N-1}{2}]$. The bijectivity of f_N ensures that x is generated uniformly over $\mathbb{Z}_N$.

In the PAC setting, a learning algorithm has to find, for every concept $g_{N,s} \in \mathcal{C}_n$, for every data distribution $\mathcal{D}$, and for all $\varepsilon, \delta \in (0, 1/2)$, a hypothesis function h that, with probability $1-\delta$, satisfies $\Pr_{x\sim\mathcal{D}}[g_{N,s}(x) \neq h(x)] \leq \varepsilon$ in time and number of samples both polynomial in n, $1/\varepsilon$ and $1/\delta$.

Note that in Ref. [125], the authors consider instead the concept class

$$\mathcal{C}'_n = \{g_{N,i} : \mathbb{Z}_N \to \{0,1\} \mid g_{N,i}(x) = \mathrm{bin}(i, g_N(x)),\ \text{for } g_N \in \mathcal{F}_n,\ i \in [n]\} \tag{6.48}$$

for any integer n, where $\mathrm{bin}(i, y)$ denotes the i-th bit of y in binary form. This concept class is also not efficiently PAC learnable, although this is only shown in a much weaker sense. The authors show that no classical

algorithm can achieve an error $\varepsilon = 1/n^2$ with failure probability $\delta = 1/n^2$ in $\mathcal{O}(\text{poly}(n))$ time, while, for the concept class $\mathcal{C}_n$ we consider, we can show that $\varepsilon = 1/2 - 1/\text{poly}(n)$ and $\delta = 1/3$ would already break the DCR assumption.

Classical hardness

To show that classical learners cannot achieve significantly better than random guesses on the class C_n, we make use of a result from Alexi *et al.*[127]. This result makes use of the notion of a $\varepsilon(n)$-oracle:

Definition 6.C.3 ($\varepsilon(n)$-oracle)
Let $O_{N,s}$ be a probabilistic oracle, such that $O_{N,s}(x)$ computes $g_{N,s}(x)$ correctly with probability $1/2 + \varepsilon(n)$ over the random choice of x and the internal randomness of the oracle. We say that $O_{N,s}$ is an $\varepsilon(n)$-oracle.

Alexi *et al.* show that a $1/\text{poly}(n)$-oracle is sufficient to break the DCR assumption.

Lemma 6.C.2 (Corollary (a) to Theorem 1 in [127])
For any $g_{N,s} \in \mathcal{C}_n$, given a $1/poly(n)$-oracle $O_{N,s}$ to $g_{N,s}$, there exists a $\mathcal{O}(poly(n))$-time algorithm that uses $O_{N,s}$ to compute $g_N(x)$, $\forall x \in \mathbb{Z}_N$ (with success probability, e.g., $9/10$).

We make use of this result to show the following lemma:

Lemma 6.C.3
Under the discrete cube root assumption, no $\mathcal{O}(poly(n))$-time classical learning algorithm can achieve an expected error

$$\Pr_{x \sim \mathcal{U}(\mathbb{Z}_N)}[h(x) \neq g_{N,s}(x)] \leq 1/2 - 1/poly(n) \tag{6.49}$$

with probability $2/3$ over the random generation of its training data and its internal randomness, and this for every concept $g_{N,s} \in \mathcal{C}_n$. $\mathcal{U}(\mathbb{Z}_N)$ is the uniform distribution over $\mathbb{Z}_N$.

Proof. Suppose by contradiction that such a learning algorithm would exist for a certain concept $g_{N,s} \in \mathcal{C}_n$. Then, given N, one can use this learning algorithm to generate with probability $2/3$ a $1/\text{poly}(n)$-oracle $O_{N,s} = h$. This is possible since the generation of training data for the concept $g_{N,s}$ is classically efficient given N. Now, by applying Lemma 6.C.2, one obtains a $\mathcal{O}(\text{poly}(n))$-time algorithm that uses this $O_{N,s}$ to compute $g_N(x)$, $\forall x \in \mathbb{Z}_N$, with success probability $9/10$. The overall success probability of this procedure, taken over the random choice of x, N and the learning algorithm is 0.6, which contradicts the DCR assumption. □

6.C.2 A simple shadow model

In this section we show how to construct a simple shadow model which can solve the same learning task for which we just showed classical hardness. This shadow model is obtained from the following flipped model:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$$
$$\rho(\boldsymbol{\theta}) = |d', s'\rangle\langle d', s'| \;\; \& \;\; O(\boldsymbol{x}) = \sum_{d', s'} g_{N,s'}(\boldsymbol{x}) \, |d', s'\rangle\langle d', s'| \, . \tag{6.50}$$

That is, $\rho(\boldsymbol{\theta})$ consists of an n-qubit register which contains the candidate key $d' \in \mathbb{Z}_N$ and a second n-qubit register containing the candidate separating $s' \in \mathbb{Z}_N$ that is used to label whether $g_N(\boldsymbol{x}) = \boldsymbol{x}^d \bmod N \in [s', s' + \frac{N-1}{2}]$.

Lemma 6.C.4
The concept class $\mathcal{C}_n$ is efficiently PAC learnable under the uniform distribution $\mathcal{U}(\mathbb{Z}_N)$ using a shadow model.

Proof. We first describe how $\rho(\boldsymbol{\theta})$ can be computed efficiently quantumly. Using the specification of N provided by the training data[7], one can use Shor's algorithm to compute the factors p, q of N with arbitrarily high probability of success. This in turn allows to compute $\phi(N) = (p-1)(q-1)$ and the key $d' = d$ that satisfies $3d = 1 \bmod \phi(N)$ using Euclid's algorithm. As for the candidate separating s', it can be encoded using Pauli-X gates. The observable $O(\boldsymbol{x})$ can also be evaluated efficiently from computational basis measurements. For an outcome (d', s'), one simply computes $g_N(\boldsymbol{x}) = \boldsymbol{x}^d \bmod N$ via modular exponentiation and checks whether the output is in $[s', s' + \frac{N-1}{2}]$.
This model is naturally specified as a shadow model. One preparation of $\rho(\boldsymbol{\theta})$ followed by a computational basis measurement results in the classical advice $\omega(\boldsymbol{\theta}) = (d', s')$. The classical evaluation $\mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta}))$ of a new data point $\boldsymbol{x}$ is done as explained in the last paragraph.
The only remaining learning aspect is to identify an s' close to s from a training set $\{(x, y_i = g_{N,s}(x))\}_{x \sim \mathcal{U}(\mathbb{Z}_N)}$. We show that a training set X of size $|X| \geq \frac{\log(\delta)}{\log(1-2\varepsilon)}$ is guaranteed to contain an x^* such that, for

[7] Strictly speaking, the PAC framework does not allow one to provide N explicitly in the training data. One way around this issue is to redefine the concepts to be of the form $\tilde{g}_{N,s} : \{0,1\} \times \mathbb{Z}_N \to \{0,1\} \mid \tilde{g}_{N,s}(0, x) = g_{N,s}(x)$ and $\tilde{g}_{N,s}(1, x) =$ "j-th bit of N, for j encoded in the first $\log(N)$ bits of x". This way, we can efficiently recover N from the uniform distribution $\mathcal{U}(\{0,1\} \times \mathbb{Z}_N)$, while only marginally impacting the training data.

$s' = g_N(x^*)$, $|s - s'| \leq \varepsilon N$ with probability $1 - \delta$. We take this x^* to be $x^* = \text{argmin}_{x \in X} \mathcal{L}(x^d \bmod N)$, for $\mathcal{L}(y) = \sum_{x \in X} |g_{N,y}(x) - g_{N,s}(x)|$ the training loss on the training set X. We show this by proving:

$$\Pr\left(|s' - s| \geq \varepsilon N\right) \leq \delta. \tag{6.51}$$

This probability is precisely the probability that no $g_N(x) \in \{g_N(x)\}_{x \in X}$ is within ε distance of s, i.e.,

$$\Pr\left(\bigcap_{x \in X} g_N(x) \notin [s - \varepsilon N, s + \varepsilon N]\right). \tag{6.52}$$

As the elements of the training set are all identically distributed, we have that this probability is equal to

$$\Pr\left(g_N(x) \notin [s - \varepsilon N, s + \varepsilon N]\right)^{|X|}. \tag{6.53}$$

Since all the datapoints are uniformly sampled from $\mathbb{Z}_N$, the probability that a datapoint is in any region of size $2\varepsilon N$ is just 2ε. With the assumption that $|X| \geq \log_{1-2\varepsilon}(\delta)$ (and assuming $\varepsilon < 1/2$), we get:

$$\Pr\left(|s' - s| \geq \varepsilon N\right) \leq (1 - 2\varepsilon)^{\log_{1-2\varepsilon}(\delta/2)} = \delta. \tag{6.54}$$

From here, we simply notice that $|s' - s| \leq \varepsilon N$ guarantees an expected error

$$\Pr_{x \sim \mathcal{U}(\mathbb{Z}_N)}[g_{N,s'}(x) \neq g_{N,s}(x)] \leq 2\varepsilon. \tag{6.55}$$

□

6.D Relations between shadow models

6.D.1 Flipped models are universal

In this section we show that any shadowfiable model as defined in Defs. 6.3.1 and 6.3.2 can be approximated by an efficiently shadowfiable flipped model. This result corresponds to Lemma 1 in the main text, and more formally in the following lemma.

Lemma 6.D.1
Let $f_{\boldsymbol{\theta}}$ be a shadowfiable model acting on n qubits as defined in Def. 6.3.2 and let $\varepsilon, \delta > 0$. There exists a flipped model $g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$ acting

on $\mathcal{O}(poly(n))$ qubits and evaluatable in $\mathcal{O}(poly(n, 1/\varepsilon))$ time such that

$$\max_{\boldsymbol{x}\in\mathcal{X}} |f_{\boldsymbol{\theta}}(\boldsymbol{x}) - g_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \varepsilon. \tag{6.56}$$

Moreover, this flipped model is also shadowfiable for the error parameter ε and the success probability $1-\delta$. More precisely, a computational basis measurement of $\rho(\boldsymbol{\theta})$ yields an advice $\omega(\boldsymbol{\theta})$ such that with probability $1-\delta$ over the randomness of this measurement, we have

$$\max_{\boldsymbol{x}\in\mathcal{X}} |f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{g}_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \varepsilon, \tag{6.57}$$

where $\widetilde{g}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta}))$ is a classical $\mathcal{O}(poly(n, 1/\varepsilon, 1/\delta, d))$-time algorithm that processes the advice $\omega(\boldsymbol{\theta})$ along with an input $\boldsymbol{x} \in \mathbb{R}^d$.

Proof. By Def. 6.3.2, the shadowfiable model $f_{\boldsymbol{\theta}}$ admits for every error of approximation $\varepsilon' > 0$ and probability of failure $\delta' > 0$ a shadow model $\widetilde{f}_{\boldsymbol{\theta}}$ that uses $m \cdot M \in \mathcal{O}(\text{poly}(n, 1/\varepsilon', 1/\delta'))$ qubits and guarantees

$$\max_{\boldsymbol{x}\in\mathcal{X}} \left| f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right| \leq \varepsilon' \tag{6.58}$$

with probability $1-\delta'$ over the generation of its advice. Out of this shadow model, we use the construction described in Fig. 6.4.b) to define the flipped model $g_{\boldsymbol{\theta}}(\boldsymbol{x})$. Since this flipped model corresponds to the evaluation of the shadow model $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ averaged over the randomness of $\omega(\boldsymbol{\theta})$, we have, for all $\boldsymbol{x} \in \mathbb{R}^d$:

$$|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - g_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \left| (1-\delta')\varepsilon' + \delta' \left\| \widetilde{f}_{\boldsymbol{\theta}} \right\| \right| \tag{6.59}$$

$$\leq \varepsilon' + \delta' \left(\left\| \widetilde{f}_{\boldsymbol{\theta}} \right\| + \varepsilon' \right) \tag{6.60}$$

where we use that with probability $1-\delta'$ we have $\left| f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right| \leq \varepsilon'$ and otherwise assume the worse case error $\left| f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right| \leq 2 \left\| \widetilde{f}_{\boldsymbol{\theta}} \right\|$, for $\left\| \widetilde{f}_{\boldsymbol{\theta}} \right\| = \max_{\boldsymbol{x}\in\mathcal{X}} \left| \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right|$ (which can also be capped to $\max_{\boldsymbol{x}\in\mathcal{X}} |f_{\boldsymbol{\theta}}(\boldsymbol{x})|$ without loss of generality). Therefore, by setting $\varepsilon' = \frac{\varepsilon}{2}$ and $\delta' = \min\left\{ \frac{\varepsilon}{2\left(\left\| \widetilde{f}_{\boldsymbol{\theta}} \right\| + \varepsilon'\right)}, \delta \right\}$ (important for the second part of the proof), we get

$$\max_{\boldsymbol{x}\in\mathcal{X}} |f_{\boldsymbol{\theta}}(\boldsymbol{x}) - g_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \varepsilon. \tag{6.61}$$

This proves the first part of the lemma. From here, it is straightforward to notice that a measurement of $\rho(\boldsymbol{\theta}) = |\omega(\boldsymbol{\theta})\rangle\langle\omega(\boldsymbol{\theta})| \otimes |0\rangle\langle 0|^{\otimes a}$ in the computational basis yields an advice $\omega(\boldsymbol{\theta})$ such that the algorithm $\mathcal{A}$ associated to the shadow model $\widetilde{f}_{\boldsymbol{\theta}}$ satisfies

$$\max_{\boldsymbol{x}\in\mathcal{X}} |f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta}))| \leq \varepsilon' < \varepsilon \tag{6.62}$$

with probability at least $1 - \delta' \geq 1 - \delta$ over the randomness of measuring the advice. □

6.D.2 BQP and P/poly

In this section we give a rigorous proof that there exist quantum models that are not shadowfiable, under the assumption that BQP $\not\subset$ P/poly. The approach we take is similar to that of Huang *et al.* [131] (Appendix A), although we consider different complexity classes and therefore show different results. Let us start by noting that the shadow models defined in Def. 6.3.1 compute functions in a subclass of P/poly, namely the subclass in which the advice $\omega(\boldsymbol{\theta})$ is efficiently generated from the measurements of a polynomial number of quantum circuits. We call this complexity class BPP/qgenpoly and it is obvious that BPP/qgenpoly $\subseteq$ P/poly as the latter is equal to BPP/poly [130] and contains all classically efficiently computable functions with advice of polynomial length, without any constraints on how the advice is generated.

On the other hand, we know that quantum models can compute all functions in BQP. To see this, we first refer the reader to the definition of BQP in Def. 6.A.4. It is easy to show that for any language L in BQP, there exists a quantum model which can decide this language. Consider the following quantum model $f_n = \mathrm{Tr}[\rho(x)O_n]$, depicted in Fig. 6.5:

$$\rho(x) = \bigotimes_{i=1}^{n} X_i^{x_i} |0\rangle\langle 0| X_i^{x_i} \quad \& \quad O_n = U_n^\dagger Z_1 U_n \tag{6.63}$$

Where X_i is the Pauli-X gate acting on the i-th qubit, here parametrized by $x_i \in \{0, 1\}$, the i-th bit of x, Z_1 is the Pauli observable on the first qubit, and U_n is the quantum circuit used to decide the language L in Def. 6.A.4. Then:

1. For all $x \in L$, $f_n(x) = \Pr[\text{the output of } U_{|x|} \text{ applied on the input } x \text{ is } 1]$ - $\Pr[\text{the output of } U_{|x|} \text{ applied to the input } x \text{ is } 0] \geq 2/3 - 1/3 = 1/3$.

2. For all $x \notin L$, $f_n(x) = \Pr[\text{the output of } U_{|x|} \text{ applied on the input } x \text{ is } 1] - \Pr[\text{the output of } U_{|x|} \text{ applied to the input } x \text{ is } 0] \leq 1/3 - 2/3 = -1/3$.

Therefore, as $f_n(x) > 0$ if $x \in L$ and $f_n(x) < 0$ if $x \notin L$ such quantum model could efficiently decide the language. We show that if all such quantum models would be shadowfiable then $\mathsf{BQP} \subseteq \mathsf{P/poly}$.

Lemma 6.D.2
If all quantum models $f_{\boldsymbol{\theta}}$ are shadowfiable with the guarantee that, $\forall x \in \mathcal{X}$,

$$\left| f_{\boldsymbol{\theta}}(x) - \widetilde{f}_{\boldsymbol{\theta}}(x) \right| < 0.15, \tag{6.64}$$

with probability at least $2/3$ over the shadowing phase and the randomness of evaluating the shadow model $\widetilde{f}_{\boldsymbol{\theta}}$, then $\mathsf{BQP} \subseteq \mathsf{P/poly}$.

Proof. Consider a language L in BQP. We showed that there exists a quantum model $f_{\boldsymbol{\theta}} = f_n$ which can determine, for every $x \in \{0,1\}^n$, whether x is in L. Now, by our assumption, there exists a shadow model $\widetilde{f}_n$ such that, for every $x \in \{0,1\}^n$, $|f_n(x) - \widetilde{f}_n(x)| < 0.15$ with probability greater than $2/3$ over the randomness of sampling the advice $\omega(\boldsymbol{\theta})$ and the (potential) internal randomness of its classical algorithm $\mathcal{A}$. To get rid of these two sources of randomness, we make use of the same proof strategy as for Adleman's theorem [130]: Consider now a new algorithm $\mathcal{A}'$ which runs $\mathcal{A}$ $18n$ times, each time using a new sampled advice string $\omega(\boldsymbol{\theta})$ and a new random bit-string for its internal randomness. Then we take a majority vote from the $18n$ runs. By Chernoff bound, the probability that for any $x \in \{0,1\}^n$ the algorithm $\mathcal{A}'$ fails to determine if x belongs to L is at most $1/e^n$. Then, by union bound, the probability that $\mathcal{A}'$ decides all the $x \in \{0,1\}^n$ correctly is at least $1 - 2^n/e^n > 0$. This implies that there exists a particular choice of the $18n$ strings $\omega(\boldsymbol{\theta})$ and $18n$ random bit-strings used in each run of the algorithm $\mathcal{A}$ such that $\mathcal{A}'$ is correct for all x. The algorithm $\mathcal{A}'$, along with this particular choice of $18n$ strings as advice (which is of size polynomial in n) is our $\mathsf{P/poly}$ algorithm. Note that it guarantees $\forall \boldsymbol{x} \in \{0,1\}^n$, $\widetilde{f}_n(x) > 0$ if $x \in L$ and $\widetilde{f}_n(x) < 0$ if $x \notin L$. Therefore we can use the sign of $\widetilde{f}_n(x)$ to determine whether $x \in L$. This implies $\mathsf{BQP} \subseteq \mathsf{P/poly}$. □

6.D.3 Shadow models beyond Fourier

Generalization to the full domain $\boldsymbol{x} \in \mathbb{R}^n$

In Section 6.3.1, we showed how the model:

$$\begin{aligned} f_{\boldsymbol{y}}(\boldsymbol{x}) &= \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})] \\ \rho(\boldsymbol{x}) = \bigotimes_{i=1}^{n} R_Y(x_i)\,|0\rangle\langle 0|\,R_Y^\dagger(x_i) \;\&\; O(\boldsymbol{y}) &= |\boldsymbol{y}\rangle\langle\boldsymbol{y}|\,. \end{aligned} \tag{6.65}$$

for $\boldsymbol{y} \in \{0,1\}^{\otimes n}$ is not efficiently shadowfiable when we restrict the domain of $\boldsymbol{x}$ to be $\{0,\pi\}^n$ as, on this domain, $f_{\boldsymbol{y}}(\boldsymbol{x}) = \delta_{\boldsymbol{x}/\pi,\boldsymbol{y}}$ plays the role of a database search oracle. We can extend this intractability result to the full domain $\boldsymbol{x} \in \mathbb{R}^n$ by that noting that more general encoding states (even more general than those in Equation 6.65) take the form

$$\rho(\boldsymbol{x}) = |\psi(\boldsymbol{x})\rangle\langle\psi(\boldsymbol{x})|\,, \quad \text{for} \quad |\psi(\boldsymbol{x})\rangle = \sum_{\boldsymbol{j}=1}^{2^n} \alpha_{\boldsymbol{j}}(\boldsymbol{x})\,|\boldsymbol{j}\rangle \tag{6.66}$$

where $\alpha_{\boldsymbol{j}}(\boldsymbol{x}) \in \mathbb{C}$ is an arbitrary amplitude associated to a computational basis state $|\boldsymbol{j}\rangle$.

Now note that when we evaluate the quantum model $f_{\boldsymbol{y}}(\boldsymbol{x})$ on a quantum computer, we do not have direct access to the expectation values $\mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})]$ it corresponds to. We rather sample an eigenvalue of $O(\boldsymbol{y})$ according to the Born rule applied to $\rho(\boldsymbol{x})$. Therefore, a single evaluation of $f_{\boldsymbol{y}}(\boldsymbol{x})$ for the encoding $\rho(\boldsymbol{x})$ defined in Equation 6.66 returns 1 with probability $|\alpha_{\boldsymbol{y}}(\boldsymbol{x})|^2$ and 0 otherwise.

Let us call $U_{\boldsymbol{y}}$ the Grover operator associated to the database search oracle that marks $\boldsymbol{y}$, i.e.,

$$U_{\boldsymbol{y}} : |\boldsymbol{j}\rangle\,|0\rangle \mapsto |\boldsymbol{j}\rangle\,|\delta_{\boldsymbol{j},\boldsymbol{y}}\rangle\,. \tag{6.67}$$

and apply it on the state $|\psi(\boldsymbol{x})\rangle\,|0\rangle$:

$$U_{\boldsymbol{y}}\,|\psi(\boldsymbol{x})\rangle\,|0\rangle = \sum_{\boldsymbol{j}=1}^{2^n} \alpha_{\boldsymbol{j}}(\boldsymbol{x})\,|\boldsymbol{j}\rangle\,|\delta_{\boldsymbol{j},\boldsymbol{y}}\rangle\,. \tag{6.68}$$

One can then notice that measuring the second register also yields 1 with probability $|\alpha_{\boldsymbol{j}}(\boldsymbol{x})|^2$ and 0 otherwise. Therefore, a single evaluation of $f_{\boldsymbol{y}}(\boldsymbol{x})$ is as powerful as querying the Grover operator oracle in superposition,

which still suffers from the same query complexity lower bound as querying it classically.

Fourier decomposition

In this subsection, we derive the Fourier-series decomposition of the model in Eq. (6.65), i.e., the frequency spectrum Ω and the Fourier coefficients $c_{\boldsymbol{\omega}}$ in the expression:

$$f_{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{\boldsymbol{\omega}\in\Omega} c_{\boldsymbol{\omega}}(\boldsymbol{y}) e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}}. \tag{6.69}$$

We start by noting that the model has a product structure, in which each (x_i, y_i) pair contributes similarly. Notably, the overlap between the i-th qubit in the state $R_Y(x_i)\,|0\rangle = \cos\left(\frac{x_i}{2}\right)|0\rangle - \sin\left(\frac{x_i}{2}\right)|1\rangle$ and the state $|y_i\rangle$ is:

$$\left|\left(\cos\left(\frac{x_i}{2}\right)\langle 0| - \sin\left(\frac{x_i}{2}\right)\langle 1|\right)|y_i\rangle\right|^2 = \cos^2\left(\frac{x_i+\pi y_i}{2}\right). \tag{6.70}$$

From the Euler decomposition of $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$, we get:

$$\cos^2\left(\frac{x_i+\pi y_i}{2}\right) = \frac{2+e^{i(x_i+\pi y_i)}+e^{-i(x_i+\pi y_i)}}{4}, \tag{6.71}$$

such that

$$f_{\boldsymbol{y}}(\boldsymbol{x}) = \prod_{i=1}^{n} \frac{2+e^{i(x_i+\pi y_i)}+e^{-i(x_i+\pi y_i)}}{4} \tag{6.72}$$

$$= \sum_{\boldsymbol{\omega}\in\{-1,0,1\}^n} \frac{1}{2^{n+|\boldsymbol{\omega}|}} e^{i(\boldsymbol{x}+\pi\boldsymbol{y})\cdot\boldsymbol{\omega}} \tag{6.73}$$

$$= \sum_{\boldsymbol{\omega}\in\{-1,0,1\}^n} \frac{e^{i\pi\boldsymbol{y}\cdot\boldsymbol{\omega}}}{2^{n+|\boldsymbol{\omega}|}} e^{i\boldsymbol{x}\cdot\boldsymbol{\omega}} \tag{6.74}$$

where $|\boldsymbol{\omega}| = \sum_{i=1}^{n} |\omega_i|$. We can therefore identify $\Omega = \{-1,0,1\}^n$ and $c_{\boldsymbol{\omega}}(\boldsymbol{y}) = \frac{e^{i\pi\boldsymbol{y}\cdot\boldsymbol{\omega}}}{2^{n+|\boldsymbol{\omega}|}}$ and note that the frequency spectrum of this model is exponentially large in n, with all its coefficients being non-zero, exponentially small in n, and differing only by a phase depending on $\boldsymbol{y}$. This justifies the exponential sample complexity needed to identify $\boldsymbol{y}$.

Obfuscation of $\boldsymbol{y}$

At this point, the interested reader might also point out that the obfuscation of the marked basis state $|\boldsymbol{y}\rangle\langle\boldsymbol{y}|$ only occurs somehow artificially by considering the observable $O(\boldsymbol{y})$ as a black-box. That is, we consider as a black-box not only the input-independent gates in the circuit but also the mapping from computational basis state to real values when measuring the output state of the circuit. More naturally, when measuring a basis state $|\boldsymbol{j}\rangle$, one would have a computable function that returns its corresponding eigenvalue, which in this case could reveal $\boldsymbol{y}$. An easy fix for this is to include the encoding of $\boldsymbol{y}$ in the circuit, by redefining $O(\boldsymbol{y})$ as

$$O(\boldsymbol{y}) = \bigotimes_{i=1}^{n} X^{y_i} |0\rangle\langle 0| X^{y_i}, \tag{6.75}$$

which delegates the obfuscation of $\boldsymbol{y}$ to gates in the circuit.

One can also go a step further in this obfuscation by considering instead the following observables

$$O(\boldsymbol{y}) = V_{\mathrm{DLP}} |\boldsymbol{y}\rangle\langle\boldsymbol{y}| V_{\mathrm{DLP}}^{\dagger} \tag{6.76}$$

where V_{DLP} is the unitary[8] that maps a basis state to its discrete logarithm:

$$V_{\mathrm{DLP}} : |\boldsymbol{y}\rangle \mapsto \left|(\log_g(\boldsymbol{y}) \bmod p) + 1\right\rangle = |\boldsymbol{y}'\rangle\,. \tag{6.77}$$

Now, even the knowledge of $\boldsymbol{y}$ and a description of the quantum circuit do not help identify $\boldsymbol{y}'$ classically, under the classical hardness assumption of DLP. Moreover, we still retain the hardness of Fourier-shadowing the resulting model $f_{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{y})]$ from the same database-search arguments. And finally, the flipped model associated to $f_{\boldsymbol{y}}$ still benefits from the same efficient shadowing procedure, as $O(\boldsymbol{y})$ can be prepared on a quantum computer and measured in the computational basis to reveal $\boldsymbol{y}'$.

6.D.4 Shadowfiablility

In our definition of shadowfiable models in the main text (see Def. 6.3.2), we take the convention that the shadow model should agree with the original quantum model for all possible inputs $\boldsymbol{x} \in \mathcal{X}$. This choice makes sense for two reasons:

[8] Note that this is indeed a unitary transformation, but that potentially needs auxiliary qubits to be implemented unitarily on a quantum computer.

1. We would like the shadowing procedure to work on all potential data distributions, as to be applicable in all learning tasks a given quantum model could be used in.

2. In the context of machine learning, one typically considers PAC conditions, meaning that the final model should achieve a small error $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}|h(\boldsymbol{x}) - g(\boldsymbol{x})|$ only with respect to some data distribution $\mathcal{D}$. Note that if the quantum model to be shadowfied achieves these PAC conditions, our demands on worst-case approximation will guarantee that the shadow model achieves them as well.

Nonetheless, one may still be interested in a notion of shadowfiability that considers an average-case error

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left|\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(x)\right| \tag{6.78}$$

with respect to a specified data distribution $\mathcal{D}$. It is not entirely clear which models can still be shadowfied in this way. But our results on the universality of flipped models (Lemma 1 in the main text and Lemma 6.D.1 in the Appendix), as well as on the existence of quantum models that are not shadowfiable (Theorem 6.2 in the main text and Lemma 6.D.2) would also hold. More precisely, for each of these results, respectively:

1. The same proof structure of Lemma 6.D.1 can be used, as the constructed flipped model only adds a small controllable error to each $\boldsymbol{x} \in \mathcal{X}$.

2. One can consider here instead of quantum models that compute arbitrary functions in BQP, a restricted model that computes (single bits of) the discrete logarithm $\log_g(\boldsymbol{x})$ mod p (analogous to our DCR concept class defined in Equation 6.46). The result of Liu *et al.* [108] (Theorem 6 in the Supplementary Information) shows the classical hardness of achieving an expected error $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}|h(\boldsymbol{x}) - g(\boldsymbol{x})| \leq 1/2 - 1/\mathrm{poly}(n)$ for such target functions (and a hypothesis h producing labels $h(\boldsymbol{x}) \in \{0, 1\}$), under the assumption that DLP $\notin$ BPP. One can then use this result to show that there exist quantum models that are not average-case shadowfiable under the assumption that DLP $\notin$ P/poly.

Algorithm 6.1: Flipped evaluation of a conventional model

Input: an n-qubit unitary $U(\boldsymbol{x})$ and a quantum state ρ_0 (diagonal in the computational basis) such that $\rho(\boldsymbol{x}) = U(\boldsymbol{x})\rho_0 U^\dagger(\boldsymbol{x})$, n-qubit unitaries $V(\boldsymbol{\theta})$ and $\{W_i\}_{1\leq i\leq d}$, real values $\{w_i\}_{i=1}^d$, $\{\lambda_{i,j}\}_{0\leq j\leq 2^n-1}^{1\leq i\leq d}$, such that $O(\boldsymbol{\theta}) = \sum_i w_i V^\dagger(\boldsymbol{\theta}) O_i V(\boldsymbol{\theta})$ with $O_i = \sum_j \lambda_{i,j} W_i^\dagger |j\rangle\langle j| W_i$.

Output: A flipped evaluation of the conventional model $\mathrm{Tr}[\rho(\boldsymbol{x})O(\boldsymbol{\theta})]$

1 Initialize $o = 0$, $N = \mathcal{O}(\|O\|_1^2/\varepsilon^2)$;
2 **for** N *iterations* **do**
3 Sample $i \in \{1, \ldots, d+1\}$ w.p. $\left(\frac{w_1\|O_1\|_1}{\|O\|_1}, \ldots, \frac{w_d\|O_d\|_1}{\|O\|_1}, \frac{\|O\|_1 - \sum_{i=1}^d w_i\|O_i\|_1}{\|O\|_1}\right)$;
4 **if** $i = d+1$ **then**
5 break iteration (or alternatively prepare $\sigma(\boldsymbol{\theta}) = I/2^{n+1}$ and jump to line 9);
6 Sample $b \in \{+, -\}$ w.p. $\frac{\|O_{i,b}\|_1}{\|O_i\|_1}$;
7 Sample $j \in \{0, \ldots, 2^n - 1\}$ w.p. $\frac{\max(0, b\lambda_{i,j})}{\|O_{i,b}\|_1}$;
8 Prepare $\sigma(\boldsymbol{\theta}) = |\widetilde{b}\rangle\langle\widetilde{b}| \otimes V^\dagger(\boldsymbol{\theta}) W_i^\dagger |j\rangle\langle j| W_i V(\boldsymbol{\theta})$, for $\widetilde{b} = 2b - 1$;
9 Measure $(I \otimes U^\dagger(\boldsymbol{x}))\sigma(\boldsymbol{\theta})(I \otimes U(\boldsymbol{x}))$ in the computational basis and call the outcome $|\widetilde{b}\rangle \otimes |j\rangle$;
10 $o \leftarrow o + b\|O\|_1 \rho_{0,j}$, where $\rho_{0,j}$ is the j-th diagonal element of ρ_0;
11 **return** o/N

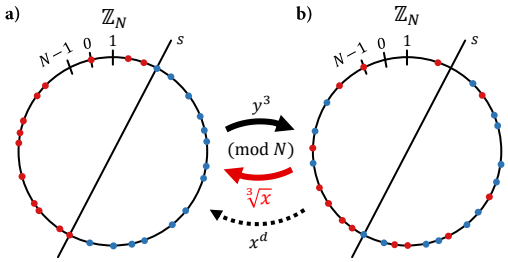


Figure 6.3: A visualization of the functions involved in the quantum advantage learning task. The core functions of this task map $\mathbb{Z}_N = \{0, \ldots, N-1\}$ to itself, for N a large semiprime. a) In feature space, data is linearly separable by a hyperplane parametrized by a certain $s \in \mathbb{Z}_N$. One can efficiently transform data y in feature space into its corresponding data x in input space via the "discrete cube" function $x = y^3 \bmod N$. b) To a fully classical learner, data in input space looks randomly labeled, as inverting it back to feature space via the discrete cube root function $y = \sqrt[3]{x} \bmod N$ is believed to be classically intractable. However, a shadow model can make use of the trap-door property of the discrete cube root function to efficiently compute a key $d \in \mathbb{Z}_N$ using a quantum computer and classically map data to feature space through the transformation $y = x^d \bmod N$.

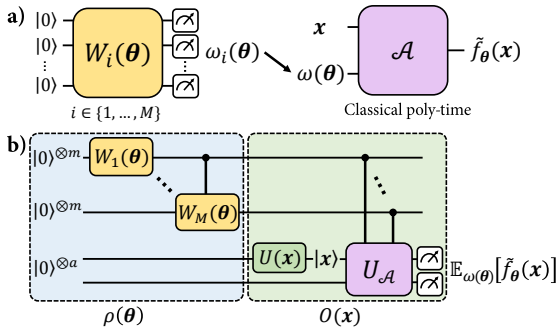


Figure 6.4: All shadow models can be expressed as shadowfiable flipped models. a) A shadow model consists of M unitary circuits $W_i(\boldsymbol{\theta})$ that can be chosen adaptively, and that generate advice $\omega_i(\theta)$ from computational basis measurements of the states $W_i(\boldsymbol{\theta})\,|0\rangle^m$. This advice, along with a (binary description of) an input $\boldsymbol{x} \in \mathbb{R}^d$ are processed by a classical algorithm $\mathcal{A}$ to compute an approximation $\widetilde{f_{\boldsymbol{\theta}}}$ of the shadowfiable model $f_{\boldsymbol{\theta}}$. b) A coherent implementation of this shadow model, where the unitaries $W_i(\boldsymbol{\theta})$ are applied on different m-qubit registers, and coherently controlled by previous registers (for adaptivity). These M registers constitute the coherent encoding of the advice $|\omega(\boldsymbol{\theta})\rangle$. The algorithm $\mathcal{A}$ can then be simulated by a reversible quantum computation $U_{\mathcal{A}}$ (see Sec. 3.2.5. in [140]) that processes a binary encoding $|\boldsymbol{x}\rangle$ of $\boldsymbol{x}$ and the coherent advice $|\omega(\boldsymbol{\theta})\rangle$ (either directly or indirectly via controlled operations that imprint $|\omega(\boldsymbol{\theta})\rangle$ on an auxiliary register). This coherent implementation of the shadow model can be viewed as a shadowfiable flipped model $g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{\theta})O(\boldsymbol{x})]$, such that one evaluation of this model samples an advice $\omega(\boldsymbol{\theta})$ and evaluates $\mathcal{A}(\boldsymbol{x}, \omega(\boldsymbol{\theta}))$ for that advice and a given $\boldsymbol{x}$.

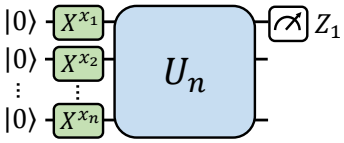


Figure 6.5: A universal quantum model for BQP. For an n-dimensional input $\boldsymbol{x} \in \{0,1\}^n$, this model acts on n qubits, encodes $\boldsymbol{x}$ in its binary form $|\boldsymbol{x}\rangle$ and applies a poly(n)-time unitary U_n before a Pauli-Z measurement of the first qubit. For appropriately chosen unitaries $\{U_n : n \in \mathbb{N}\}$, this model can decide any language in BQP. For more general computational basis measurements, the resulting model can represent arbitrary functions in FBQP, the functional version of BQP.

CHAPTER 7

Conclusion

In this chapter we present the conclusions of the thesis. First, we will recall the key research questions and detail the answers. Second, this chapter will provide details on the limitations of the methods discussed. Finally, we will reoutline promising directions for future work.

7.1 Research Overview

This thesis has centred on how we can differentiate classical and quantum computing, either practically or theoretically. The first of the research questions given in the introduction tackled the theoretical aspect of this:

Research question 1
What is the strongest theoretical basis for the claim "In polynomial time quantum computers can perform computations that classical computers cannot"?

This was the question addressed by the first two chapters, in chapter 2 we addressed this question directly by casting it into a comparison between FBQP and FBPP. Chapter 2 strengthened the separation between these classes to a new state of the art, i.e. that they are separate unless the polynomial hierarchy collapses to the 2nd level or two conjectures about permanents turn out to be incorrect. The chapter discussed the significance

of this improvement, as the separation between FBQP and FBPP is in some sense the most rigorous separation of quantum and classical computing currently known.

Chapter 3 answered a similar research question as chapter 2, but now in the setting of advice.

Research question 2
If polynomial-time quantum computers can give better advice than polynomial-time classical computers, can we find such an advice-generating algorithm?

By focusing on the question of bounded advice, we found that if a quantum-classical separation could be reduced to advice then quantum and classical computing could not be separated in exponential time. This concept was put to the test in chapter 6, which developed a machine learning algorithm capable of exploiting any advantage posed by quantum advice. This algorithm proved successful both practically and for trap-door functions.

The remainder of the chapters focused on a more practical question than the first two. Instead of asking about the improvements offered by quantum computing in the limit of growing size, we asked how these advantages could be gained earlier. In particular, if we can lower the burden of running a particular algorithm on a smaller quantum computer.

Research question 3
Do there exist methods to reduce the number of qubits required to run a given machine learning algorithm?

In chapter 4 we demonstrated one potential algorithm that was able to replicate the performance of a higher-qubit-count machine learning algorithm with fewer qubits, suggesting the question may resolve to yes. Chapter 5 demonstrated the limitations of this scheme and all other cut-local schemes, this chapter proved that as long as $\mathsf{BQP} \neq \mathsf{BPP}$ then the form of circuit cutting used in the preceding chapter could never successfully cut **all** circuits.

7.1.1 Future work

The chapter on the separation of FBQP and FBPP highlighted further improvements, particularly lowering the collapse of the polynomial hierarchy even further. In the similar case of **exactly sampling** from a quantum computer Fuji et al. [21] were able to collapse the hierarchy to $\mathsf{AM} \cap \mathsf{coAM}$ if the task was possible classically. While their techniques did not extend

to the practically relevant case captured by SampBQP, they still provide information on what might be possible with further work. Indeed, our collapse is a much better starting point for further collapses.

For advice and advice following/giving models, it would be interesting to see where the advice classes could be deployed. While the surrogate model proposed by chapter 6 is in some sense optimal up to a polynomial factor this polynomial factor could be large. It would therefore be interesting to see other advice-following/generating models be developed and to compare how these perform in real-world tasks. Further work stemming from chapter 3 could focus further on developing the connections between bounded advice classes, as opposed to linking back to other classes as we did.

While chapter 5 proved that cut-local schemes would never be able to remove even a single qubit efficiently, there is still work to be done to generalise this result. While a fully general theorem is essentially the field of circuit complexity it may be possible to produce better bounds on the ability of more complicated circuit-cutting algorithms. This type of research will naturally also suggest how a more efficient machine learning algorithm could be developed to combine multiple small quantum circuits into outputs that would traditionally need larger circuits.

Bibliography

[1] M. A. Nielsen & I. Chuang. Quantum computation and quantum information (2002).

[2] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, **2**, 79 (2018).

[3] S. Bravyi, G. Smith & J. A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, **6**, 021043 (2016).

[4] T. Peng, A. W. Harrow, M. Ozols & X. Wu. Simulating large quantum circuits on a small quantum computer. *Physical Review Letters*, **125** (2020).

[5] K. Mitarai & K. Fujii. Constructing a virtual two-qubit gate by sampling single-qubit operations. *New Journal of Physics*, **23**, 023021 (2021).

[6] A. Eddins, M. Motta, T. P. Gujarati, S. Bravyi, A. Mezzacapo, C. Hadfield & S. Sheldon. Doubling the size of quantum simulators by entanglement forging. *PRX Quantum*, **3**, 010309 (2022).

[7] C. Piveteau & D. Sutter. Circuit knitting with classical communication. *arXiv preprint arXiv:2205.00016* (2022).

[8] S. C. Marshall, C. Gyurik & V. Dunjko. High dimensional quantum machine learning with small quantum computers. *Quantum*, **7**, 1078 (2023).

[9] A. Lowe, M. Medvidović, A. Hayes, L. J. O'Riordan, T. R. Bromley, J. M. Arrazola & N. Killoran. Fast quantum circuit cutting with randomized measurements. *Quantum*, **7**, 934 (2023).

[10] H. Harada, K. Wada & N. Yamamoto. Optimal parallel wire cutting without ancilla qubits, arXiv:2303.07340 (2023).

[11] E. Pednault. An alternative approach to optimal wire cutting without ancilla qubits, arXiv:2303.08287 (2023).

[12] S. Bravyi, G. Smith & J. A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, **6**, 021043 (2016).

[13] M. Sipser. Introduction to the theory of computation. *ACM Sigact News*, **27**, 27–29 (1996).

[14] S. Aaronson. The equivalence of sampling and searching. *Theory of Computing Systems*, **55**, 281–298 (2014).

[15] S. Aaronson & A. Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342 (2011). URL `https://doi.org/10.1145/1993636.1993682`.

[16] T. Baker, J. Gill & R. Solovay. Relativizations of the p=?np question. *SIAM Journal on computing*, **4**, 431–442 (1975).

[17] S. Aaronson & A. Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342 (2011).

[18] M. J. Bremner, A. Montanaro & D. J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical review letters*, **117**, 080501 (2016).

[19] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **461**, 3473–3482 (2005).

[20] R. O'Donnell & A. C. Say. The weakness of ctc qubits and the power of approximate counting. *ACM Transactions on Computation Theory (TOCT)*, **10**, 1–22 (2018).

[21] K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate & S. Tani. Power of quantum computation with few clean qubits. *arXiv preprint arXiv:1509.07276* (2015).

[22] S. Aaronson, G. Kuperberg & C. Granade. The complexity zoo (2005).

[23] J. Feigenbaum & L. Fortnow. On the random-self-reducibility of complete sets, university of chicago technical report 90-22. *Computer Science Department*, **20** (1990).

[24] V. Arvind & J. Köbler. New lowness results for zppnp and other complexity classes. *Journal of Computer and System Sciences*, **65**, 257–277 (2002).

[25] S. Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, **20**, 865–877 (1991).

[26] M. Blum & S. Kannan. Designing programs that check their work. *Journal of the ACM (JACM)*, **42**, 269–291 (1995).

[27] J. Köbler & O. Watanabe. New collapse consequences of np having small circuits. *SIAM Journal on Computing*, **28**, 311–324 (1998).

[28] Y. Han, L. A. Hemaspaandra & T. Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, **26**, 59–78 (1997).

[29] T. Morimae. Hardness of classically sampling the one-clean-qubit model with constant total variation distance error. *Physical Review A*, **96**, 040302 (2017).

[30] D. Aharonov, X. Gao, Z. Landau, Y. Liu & U. Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 945–957 (2023).

[31] L. Stockmeyer. On approximation algorithms for# p. *SIAM Journal on Computing*, **14**, 849–861 (1985).

[32] L. G. Valiant & V. V. Vazirani. Np is as easy as detecting unique solutions. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 458–463 (1985).

[33] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman et al. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17 (2015).

[34] S. Aaronson. Nsa in p/poly: The power of precomputation (2015). URL `https://scottaaronson.blog/?p=2293`. Published on May 22nd, 2015.

[35] S. Jerbi, C. Gyurik, S. C. Marshall, R. Molteni & V. Dunjko. Shadows of quantum machine learning. *Nature Communications*, **15**, 5676 (2024).

[36] F. J. Schreiber, J. Eisert & J. J. Meyer. Classical surrogates for quantum learning models. *Physical Review Letters*, **131**, 100803 (2023).

[37] J. Landman, S. Thabet, C. Dalyac, H. Mhiri & E. Kashefi. Classically approximating variational quantum machine learning with random fourier features. *arXiv preprint arXiv:2210.13200* (2022).

[38] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven & J. R. McClean. Power of data in quantum machine learning. *Nature communications*, **12**, 2631 (2021).

[39] R. Gavalda. The complexity of learning with queries. In *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*, pages 324–337 (1994).

[40] I. Volkovich. On learning, lower bounds and (un) keeping promises. In *International Colloquium on Automata, Languages, and Programming*, pages 1027–1038. Springer (2014).

[41] N. Rajgopal & R. Santhanam. On the structure of learnability beyond P/Poly. In M. Wootters & L. Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). Sections 2.4, 3.

[42] R. Gavalda. Bounding the complexity of advice functions. *Journal of Computer and System Sciences*, **50**, 468–475 (1995).

[43] K.-I. Ko & U. Schöning. On circuit-size complexity and the low hierarchy in np. *SIAM Journal on Computing*, **14**, 41–51 (1985).

[44] J. Köbler & T. Thierauf. Complexity-restricted advice functions. *SIAM Journal on Computing*, **23**, 261–275 (1994).

[45] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan & C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, **52**, 421–433 (1996).

[46] S. Aaronson & A. Drucker. A full characterization of quantum advice. *SIAM Journal on Computing*, **43**, 1131–1183 (2014).

[47] S. Aaronson, G. Kuperberg & O. Habryka. Complexity zoo (2023). URL `https://complexityzoo.net/`. Accessed: date.

[48] A. Meduna. *Turing Transducers*, pages 833–887. Springer London, London (2000). ISBN 978-1-4471-0501-5. URL `https://doi.org/10.1007/978-1-4471-0501-5_11`.

[49] S. Aaronson, H. Buhrman & W. Kretschmer. A qubit, a coin, and an advice string walk into a relational problem. *arXiv preprint arXiv:2302.10332* (2023).

[50] J. Hartmanis. On sparse sets in np–p. *Information Processing Letters*, **16**, 55–60 (1983).

[51] M. Cerezo, M. Larocca, D. García-Martín, N. Diaz, P. Braccia, E. Fontana et al. Does provable absence of barren plateaus imply classical simulability? or, why we need to rethink variational quantum computing. *arXiv preprint arXiv:2312.09121* (2023).

[52] J. Watrous. Quantum computational complexity. *arXiv preprint arXiv:0804.3401* (2008).

[53] J. Preskill. Quantum computing in the nisq era and beyond. *Quantum*, **2**, 79 (2018).

[54] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow & J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, **567**, 209–212 (2019).

[55] M. Schuld & N. Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, **122**, 040504 (2019).

[56] J.-G. Liu & L. Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, **98**, 062324 (2018).

[57] T. Peng, A. W. Harrow, M. Ozols & X. Wu. Simulating large quantum circuits on a small quantum computer. *Physical Review Letters*, **125**, 150504 (2020).

[58] W. Tang, T. Tomesh, M. Suchara, J. Larson & M. Martonosi. Cutqc: using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 473–486 (2021).

[59] M. A. Perlin, Z. H. Saleem, M. Suchara & J. C. Osborn. Quantum circuit cutting with maximum-likelihood tomography. *npj Quantum Information*, **7**, 1–8 (2021).

[60] Z. H. Saleem, T. Tomesh, M. A. Perlin, P. Gokhale & M. Suchara. Quantum divide and conquer for combinatorial optimization and distributed computing. *arXiv preprint arXiv:2107.07532* (2021).

[61] M. Bechtold. *Bringing the concepts of virtualization to gate-based quantum computing*. Master's thesis, University of Stuttgart (2021).

[62] J. Avron, O. Casper & I. Rozen. Quantum advantage and noise reduction in distributed quantum computing. *Physical Review A*, **104**, 052404 (2021).

[63] S. Basu, A. Saha, A. Chakrabarti & S. Sur-Kolay. *i*-qer: An intelligent approach towards quantum error reduction. *arXiv preprint arXiv:2110.06347* (2021).

[64] T. Haug, C. N. Self & M. S. Kim. Quantum machine learning of large datasets using randomized measurements. *Machine Learning: Science and Technology*, **4**, 015005 (2023).

[65] W. Li, S. Lu & D.-L. Deng. Quantum federated learning through blind quantum computing. *Science China Physics, Mechanics & Astronomy*, **64**, 1–8 (2021).

[66] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain & G. N. Perdue. Machine learning of high dimensional data on a noisy quantum processor. *npj Quantum Information*, **7**, 1–5 (2021).

[67] K. Fujii, K. Mizuta, H. Ueda, K. Mitarai, W. Mizukami & Y. O. Nakagawa. Deep variational quantum eigensolver: a divide-and-conquer method for solving a larger problem with smaller size quantum computers. *PRX Quantum*, **3**, 010346 (2022).

[68] X. Yuan, J. Sun, J. Liu, Q. Zhao & Y. Zhou. Quantum simulation with hybrid tensor networks. *Physical Review Letters*, **127**, 040501 (2021).

[69] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow & J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, **549**, 242–246 (2017).

[70] S. Balakrishnan & R. Sankaranarayanan. Operator-schmidt decomposition and the geometrical edges of two-qubit gates. *Quantum Information Processing*, **10**, 449–461 (2011).

[71] K. Mitarai & K. Fujii. Overhead for simulating a non-local channel with local channels by quasiprobability sampling. *Quantum*, **5**, 388 (2021).

[72] S. Jerbi, C. Gyurik, S. Marshall, H. J. Briegel & V. Dunjko. Variational quantum policies for reinforcement learning. *arXiv preprint arXiv:2103.05577* (2021).

[73] M. Mohri, A. Rostamizadeh & A. Talwalkar. *Foundations of machine learning*. MIT press (2018).

[74] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert & R. Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *arXiv preprint arXiv:2106.03880* (2021).

[75] M. Schuld, R. Sweke & J. J. Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, **103**, 032430 (2021).

[76] F. J. Gil Vidal & D. O. Theis. Input redundancy for parameterized quantum circuits. *Frontiers in Physics*, **8**, 297 (2020).

[77] P. L. Bartlett & S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, **3**, 463–482 (2002).

[78] js21 (https://mathoverflow.net/users/21724/js21). Product of estimates of mean values - concentration of measure inequality. MathOverflow. URL `https://mathoverflow.net/q/286787`. URL:https://mathoverflow.net/q/286787 (version: 2017-11-23).

[79] G. E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311* (2019).

[80] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush & H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, **9**, 1–6 (2018).

[81] S. H. Sack, R. A. Medina, A. A. Michailidis, R. Kueng & M. Serbyn. Avoiding barren plateaus using classical shadows. *arXiv preprint arXiv:2201.08194* (2022).

[82] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, **151**, 3–34 (2015).

[83] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, **29**, 141–142 (2012).

[84] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven & J. R. McClean. Power of data in quantum machine learning. *Nature Communications*, **12** (2021).

[85] E. Farhi & H. Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002* (2018).

[86] L. Yann. If you listen carefully, you can hear the goat screaming "MNIST results!" But the dude isn't listening carefully. *Twitter* (2022).

[87] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar & M. Troyer. Defining and detecting quantum speedup. *Science*, **345**, 420–424 (2014).

[88] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis & H. Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, **14**, 595–600 (2018).

[89] G. Chiribella, G. M. D'Ariano & P. Perinotti. Quantum circuit architecture. *Physical Review Letters*, **101**, 060401 (2008).

[90] N.-H. Chia, C.-N. Chou, J. Zhang & R. Zhang. Quantum meets the minimum circuit size problem. *arXiv preprint arXiv:2108.03171* (2021).

[91] E. Knill & R. Laflamme. Power of one bit of quantum information. *Physical Review Letters*, **81**, 5672 (1998).

[92] K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate & S. Tani. Impossibility of classically simulating one-clean-qubit model with multiplicative error. *Physical Review Letters*, **120**, 200502 (2018).

[93] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe & S. Lloyd. Quantum machine learning. *Nature*, **549**, 195–202 (2017).

[94] V. Dunjko & H. J. Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, **81**, 074001 (2018).

[95] M. Schuld & F. Petruccione. *Supervised learning with quantum computers*, volume 17. Springer (2018).

[96] M. Benedetti, E. Lloyd, S. Sack & M. Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, **4**, 043001 (2019).

[97] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii et al. Variational quantum algorithms. *Nature Reviews Physics*, **3**, 625–644 (2021).

[98] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, **94**, 015004 (2022).

[99] M. Schuld & N. Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, **122**, 040504 (2019).

[100] M. Schuld, A. Bocharov, K. M. Svore & N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, **101**, 032308 (2020).

[101] J.-G. Liu & L. Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, **98**, 062324 (2018).

[102] S. Jerbi, C. Gyurik, S. Marshall, H. Briegel & V. Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, **34** (2021).

[103] A. Skolik, S. Jerbi & V. Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, **6**, 720 (2022).

[104] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow & J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, **567**, 209–212 (2019).

[105] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete et al. Training of quantum circuits on a hybrid quantum computer. *Science advances*, **5**, eaaw9918 (2019).

[106] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain & G. N. Perdue. Machine learning of high dimensional data on a noisy quantum processor. *npj Quantum Information*, **7**, 161 (2021).

[107] T. Haug, C. N. Self & M. Kim. Quantum machine learning of large datasets using randomized measurements. *Machine Learning: Science and Technology*, **4**, 015005 (2023).

[108] Y. Liu, S. Arunachalam & K. Temme. A rigorous and robust quantum speed-up in supervised machine learning. *arXiv preprint arXiv:2010.02174* (2020).

[109] C. Gyurik & V. Dunjko. On establishing learning separations between classical and quantum machine learning with classical data. *arXiv preprint arXiv:2208.06339* (2022).

[110] C. Gyurik & V. Dunjko. Exponential separations between classical and quantum learners. *arXiv preprint arXiv:2306.16028* (2023).

[111] F. J. Schreiber, J. Eisert & J. J. Meyer. Classical surrogates for quantum learning models. *Physical Review Letters*, **131**, 100803 (2023).

[112] J. Landman, S. Thabet, C. Dalyac, H. Mhiri & E. Kashefi. Classically approximating variational quantum machine learning with random fourier features. *arXiv preprint arXiv:2210.13200* (2022).

[113] M. Schuld, R. Sweke & J. J. Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, **103**, 032430 (2021).

[114] S. Jerbi, L. J. Fiderer, H. Poulsen Nautrup, J. M. Kübler, H. J. Briegel & V. Dunjko. Quantum machine learning beyond kernel methods. *Nature Communications*, **14**, 517 (2023).

[115] H.-Y. Huang, R. Kueng & J. Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, **16**, 1050–1057 (2020).

[116] C. Bertoni, J. Haferkamp, M. Hinsche, M. Ioannou, J. Eisert & H. Pashayan. Shallow shadows: Expectation estimation using low-depth random clifford circuits. *arXiv preprint arXiv:2209.12924* (2022).

[117] K. Wan, W. J. Huggins, J. Lee & R. Babbush. Matchgate shadows for fermionic quantum simulation. *Communications in Mathematical Physics*, pages 1–72 (2023).

[118] H.-Y. Hu, S. Choi & Y.-Z. You. Classical shadow tomography with locally scrambled quantum dynamics. *Physical Review Research*, **5**, 023027 (2023).

[119] M. Schuld. Supervised quantum machine learning models are kernel methods. *arXiv:2101.11020* (2021).

[120] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin & Y.-K. Liu. Efficient quantum state tomography. *Nature communications*, **1**, 149 (2010).

[121] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert & R. Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum*, **5**, 582 (2021).

[122] C. Gyurik, V. Dunjko et al. Structural risk minimization for quantum linear classifiers. *Quantum*, **7**, 893 (2023).

[123] R. A. Servedio & S. J. Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, **33**, 1067–1092 (2004).

[124] R. Sweke, J.-P. Seifert, D. Hangleiter & J. Eisert. On the quantum versus classical learnability of discrete distributions. *Quantum*, **5**, 417 (2021).

[125] M. J. Kearns & U. Vazirani. *An introduction to computational learning theory*. MIT press (1994).

[126] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, **41**, 303–332 (1999).

[127] W. Alexi, B. Chor, O. Goldreich & C. P. Schnorr. Rsa and rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, **17**, 194–209 (1988).

[128] B. Casas & A. Cervera-Lierta. Multidimensional fourier series with quantum circuits. *Physical Review A*, **107**, 062612 (2023).

[129] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219 (1996). URL `https://doi.org/10.1145/237814.237866`.

[130] L. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 75–83. IEEE Computer Society (1978). URL `https://doi.org/10.1109/SFCS.1978.37`.

[131] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven & J. R. McClean. Power of data in quantum machine learning. *Nature Communications*, **12**, 1–9 (2021).

[132] L. Leone, S. F. Oliviero & A. Hamma. Nonstabilizerness determining the hardness of direct fidelity estimation. *Physical Review A*, **107**, 022429 (2023).

[133] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni et al. Quantum advantage in learning from experiments. *Science*, **376**, 1182–1186 (2022).

[134] P. Dagum, R. Karp, M. Luby & S. Ross. An optimal algorithm for monte carlo estimation. *SIAM Journal on computing*, **29**, 1484–1496 (2000).

[135] R. Canetti, G. Even & O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, **53**, 17–25 (1995).

[136] S. Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **463**, 3089–3114 (2007).

[137] M. Anthony & P. L. Bartlett. Function learning from interpolation. *Combinatorics, Probability and Computing*, **9**, 213–225 (2000).

[138] A. Ambainis, A. Nayak, A. Ta-Shma & U. Vazirani. Dense quantum coding and quantum finite automata. *Journal of the ACM (JACM)*, **49**, 496–511 (2002).

[139] L. Wang & X. Ma. Bounds of graph energy in terms of vertex cover number. *Linear Algebra and its Applications*, **517**, 207–216 (2017).

[140] M. A. Nielsen & I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press (2000).

[141] S. C. Marshall, S. Aaronson & V. Dunjko. Improved separation between quantum and classical computers for sampling and functional tasks. *arXiv preprint arXiv:2410.20935* (2024).

[142] S. C. Marshall, C. Gyurik & V. Dunjko. On bounded advice classes. *arXiv preprint arXiv:2405.18155* (2024).

[143] S. C. Marshall, C. Gyurik & V. Dunjko. High Dimensional Quantum Machine Learning With Small Quantum Computers. *Quantum*, **7**, 1078 (2023).

[144] S. C. Marshall, J. Tura & V. Dunjko. All this for one qubit? bounds on local circuit cutting schemes. *arXiv preprint arXiv:2303.13422* (2023).

[145] S. C. Marshall & J. H. Kirchner. Understanding polysemanticity in neural networks through coding theory. *arXiv preprint arXiv:2401.17975* (2024).

[146] S. Jerbi, C. Gyurik, S. C. Marshall, H. Briegel & V. Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, **34**, 28362–28375 (2021).

[147] S. Aaronson, S. Grewal, V. Iyer, S. C. Marshall & R. Ramachandran. PDQMA = DQMA = NEXP: QMA with hidden variables and non-collapsing measurements. *arXiv preprint arXiv:2403.02543* (2024).

Acknowledgments

First, I owe the greatest debt to my family. I thank my parents, Richard and Linda Marshall, for all their time and patience spent nurturing my inquisitiveness; my siblings, Helen, Andrew and Tonto for their tolerance; and my grandparents, Douglas and Ann Flower and Edwin and Flo Marshall, for so much. I also thank Redvers Flower, James Stewart, Catherine McLean and Mabel Jacques, whose procreative ability ensured the survival of the requisite genetic information for my creation.

Throughout my academic life, I have been fortunate to be mentored by a number of people for reasons that are still not entirely apparent to me. George Knee, Dominic Bradford, Jamie Friel, James Sprittles, and Scott Aaronson, all of whom gave me an amount of their valuable time far beyond what should have been logically afforded. Other people, such as Jan Kirchner and Beth Barnes, have afforded me time as part of a larger plot to achieve their own (albeit altruistic) goals, but I am non-the-less thankful for their contribution.

To Emiel and Kshiti, thank you for the day in and day out companionship. More waking hours are spent in the office than any other room, and you two have made that place an actual joy. Emiel, thank you for your entertainment. Kshiti, sorry about all the entertainment.

To Sofiene, Casper, and Yash, thank you for all your help early on in my PhD. To Riccardo, I thoroughly enjoyed our collaborations, even if you have yet to believe in your abilities as much as I believe in them.

I have often found that our research group, AQA, is the envy of other PhD students. This envy is well placed; in AQA I have found mentorship,

comradeship and friendship. I want to thank the group leaders, Vedran, Evert, Jordi, Alfons, Hao and Anna for creating such a welcoming and intellectually open community and all the members. A special thanks must be given to Alice, David, Jan, Eloïc, Elliot, Mahtab, and Ilse and the king among men, Felix, but all of AQA have helped so much over these years, I don't know half of you as well as I should like and I like less than half of you half as well as you deserve.

Moving to this country in October 2020 meant that it was always going to be a pandemic-shaped challenge. I could not have finished this PhD without the support of my friends. To Jo and Stefano, thank you for being there for me for 4 straight years. Jo, I do not understand why you haven't left yet. Fed, thank you for all the love and support, and for putting up with me when I'm in bad moods. Sarah, thank you for your patience and support through my first years. Finn, thank you for your irreplaceable friendship. I am grateful to Robert Harling, Anne le Roux, Charlotte Siegman, Toby Tremlett, Zakee Ulhaq, Janvi Ahuja, Amanda Matthes, and Jan Werner, my life trajectory would have been significantly worse without your guidance. Thank you to Jack Green, Thomas Bingham, Conrad Bullows-Weeks, Gaia Webb, Jeremy Rubinoff, Sarah Cogan, Dr Henry Israel and Jeanne Vliet, without your support when I needed it I don't think I would have finished this PhD. I offer no thanks to Adam Howes, although much is deserved. To the rest of my friends: if I forgot to name you, assume you're part of the emotional support ensemble and take full credit anyway.

Finally, I will end by thanking my mentor, Vedran Dunjko, while our relationship has not always been easy I can say with certainty that I would not be half the mathematician I am today if you had not pushed me to be so and I am very glad I chose to become your PhD student 5 years ago.

Summary

This thesis has sought to probe or prove the supposed advantages of quantum computing, first by analysing under what conditions an advantage may theoretically be present, and then by developing and scrutinising techniques to produce a practical advantage.

The introductory chapter introduces the core concepts needed to understand what quantum computing is, how it fits into concepts like complexity theory or machine learning, as well as more developed quantum computing topics, such as circuit cutting.

The second chapter develops the strongest evidence given so far that quantum computers can sample from distributions or compute functions that classical computers cannot, which we formalise as $\mathsf{SampP} \stackrel{?}{=} \mathsf{SampBQP}$ or $\mathsf{FBQP} \stackrel{?}{=} \mathsf{FBPP}$. This chapter iterates on existing conditions for the separation of quantum and classical computation, finding stronger conditions than had previously been achieved and suggesting that if quantum computers cannot provide speed-ups over classical, then widely held beliefs about complexity theory would be wrong. This progress is made by demonstrating that exact counting and approximate counting have to be different unless the polynomial hierarchy collapses to its second level.

Chapter 3 follows the lead of Chapter 2 by questioning if advice may change the separation of classical and quantum computation. Motivated by papers that identify tasks which have a likely quantum advantage only need a quantum advantage to prepare samples, such as [38]. This chapter first generalised this question, to an advice generating Turing machine

and an advice recieving machine. This generalisation allowed for general rules, particularly $\mathsf{P}/\mathsf{poly}^{\mathsf{B}} = \mathsf{P}^{\mathsf{Un}(\mathsf{B})}$. This result reveal that advice from quantum computers enhances classical computers, unless $\mathsf{BQEXP}{=}\mathsf{BPEXP}$ (if quantum and classical machines are equally useful if they are both allowed to run for exponentially long). This chapter highlighted the practical implications of quantum advice givers, as quantum computers will initially be very expensive being able to use them for a small portion of the total compute of some quantum-enhanced task will be key.

Chapters 4 and 6 can be seen as building on the practical use case of the previous chapter, by both attempting to enhance limited modern machines. Chapter 6 does this directly by developing a model which uses a quantum machine to prepare/train a model that can then be deployed using only classical hardware. The model is shown to be universal for models prepared quantumly and deployed classically, therefore the results of Chapter 3 imply it is more powerful than classical models unless $\mathsf{BQEXP}{=}\mathsf{BPEXP}$. Chapter 4 enhances limited modern machines via circuit cutting, where a model is developed that in the limit becomes the cut-up version of a parameterised quantum circuit but can also be set to use much less resources than perfect circuit cutting would require. Through experiment, it is demonstrated that this model is capable of learning distributions which are non-trivial for a quantum machine (such as handwriting), while still retaining advantages from the quantum machine (such as being able to fit quantum circuits). Both of these chapters probe the power of quantum computing by opening up the ability for us to study quantum algorithms that we would otherwise not be able to study on modern machines.

Chapter 5 takes a different tone than the previous chapters: Instead of furthering a research line to show that quantum computing is more powerful than classical computing, it suggests that a branch of research, circuit cutting, can never provide a meaningful speed up. This is achieved by showing that if circuit cutting could be done efficiently, then $\mathsf{BQP} = \mathsf{BPP}$. This has direct implications for the results of Chapter 4, showing this model can never be as good as a full-sized model for all tasks. It also helps to bound and inform a number of improvements to cutting schemes.

All of these chapters either demonstrate quantum-classical separations or provide results informative to those trying to demonstrate such a connection.

Samenvatting

Dit proefschrift onderzoekt en bewijst waar mogelijk de veronderstelde voordelen van quantum computing, eerst door te analyseren onder welke omstandigheden een voordeel theoretisch aanwezig kan zijn, en vervolgens door technieken te ontwikkelen en te onderzoeken om een praktisch voordeel te produceren.

Het inleidende hoofdstuk introduceert de kernconcepten die nodig zijn om te begrijpen wat quantum computing is, hoe het zich verhoudt tot concepten zoals complexiteitstheorie en machine learning, evenals geavanceerdere onderwerpen binnen quantum computing, zoals circuit cutting.

Het tweede hoofdstuk ontwikkelt het sterkste bewijs tot nu toe dat quantumcomputers kunnen bemonsteren uit verdelingen of functies kunnen berekenen die klassieke computers niet kunnen, wat we formaliseren als $\mathsf{SampP} \stackrel{?}{=} \mathsf{SampBQP}$ of $\mathsf{FBQP} \stackrel{?}{=} \mathsf{FBPP}$. Dit hoofdstuk herhaalt bestaande voorwaarden voor de scheiding van quantum en klassieke computing, vindt sterkere voorwaarden dan eerder bereikt en suggereert dat als quantumcomputers geen snelheidsvoordelen bieden ten opzichte van klassieke computers, zouden wijdverbreide overtuigingen over de complexiteitstheorie onjuist zijn. Deze vooruitgang wordt geboekt door aan te tonen dat exacte telling en benaderende telling verschillend moeten zijn, tenzij de polynomiale hiërarchie instort tot het tweede niveau.

Hoofdstuk 3 bouwt voort op Hoofdstuk 2 door te onderzoeken of advies de scheiding tussen klassieke en kwantumberekeningen kan beïnvloeden. Geïnspireerd door artikelen die taken identificeren die waarschijnlijk een kwantumvoordeel hebben, hebben ze alleen een kwantumvoordeel nodig

om samples voor te bereiden, zoals [38]. Dit hoofdstuk generaliseerde deze vraag eerst naar een adviesgenererende Turing-machine en een adviesontvangende machine. Deze generalisatie stond algemene regels toe, met name $\mathsf{P}/\mathsf{poly}^{\mathsf{B}} = \mathsf{P}^{\mathsf{Un(B)}}$. Dit resultaat laat zien dat advies van kwantumcomputers klassieke computers kan versterken, tenzij BQEXP=BPEXP (als kwantum- en klassieke machines even nuttig zijn als ze beide exponentieel lang mogen draaien). Dit hoofdstuk benadrukt de praktische implicaties van kwantumadviesgevers. Aangezien kwantumcomputers in eerste instantie erg duur zullen zijn, zal het belangrijk zijn om ze te kunnen gebruiken voor een klein deel van de totale berekening van een kwantumverbeterde taak.

Hoofdstukken 4 en 6 kunnen worden gezien als voortbouwend op het praktische gebruiksvoorbeeld van het vorige hoofdstuk, door beide te proberen beperkte moderne machines te verbeteren. Hoofdstuk 6 doet dit rechtstreeks door een model te ontwikkelen waarin een kwantummachine wordt gebruikt om een model voor te bereiden en te trainen dat vervolgens kan worden geïmplementeerd met alleen klassieke hardware. Het model blijkt universeel te zijn voor modellen die kwantum zijn voorbereid en klassiek zijn geïmplementeerd, daarom impliceren de resultaten van hoofdstuk 3 dat het krachtiger is dan klassieke modellen, tenzij BQEXP=BPEXP. Hoofdstuk 4 verbetert beperkte moderne machines door circuitsnijden toe te passen, waarbij een model wordt ontwikkeld dat uiteindelijk de opgeknipte versie van een geparametriseerd kwantumcircuit wordt, maar ook kan worden ingesteld om veel minder bronnen te gebruiken dan perfect circuitsnijden zou vereisen. Door experimenten wordt aangetoond dat dit model in staat is om distributies te leren die niet triviaal zijn voor een quantummachine (zoals handschrift), terwijl het nog steeds voordelen van de quantummachine behoudt (zoals het kunnen passen van quantumcircuits). Beide hoofdstukken onderzoeken de kracht van quantumcomputing door ons de mogelijkheid te bieden om quantumalgoritmen te bestuderen die we anders niet op moderne machines zouden kunnen bestuderen.

Hoofdstuk 5 heeft een andere toon dan de vorige hoofdstukken: in plaats van een onderzoekslijn voort te zetten om aan te tonen dat quantumcomputing krachtiger is dan klassieke computing, suggereert het dat een onderzoeksrichting, circuitsnijden, nooit een significante snelheidsverbetering kan opleveren. Dit wordt bereikt door aan te tonen dat als circuit cutting efficiënt zou kunnen worden gedaan, dan $\mathsf{BQP} = \mathsf{BPP}$. Dit heeft directe implicaties voor de resultaten van hoofdstuk 4, en laat zien dat dit model nooit zo goed kan zijn als een model op ware grootte voor alle taken. Het helpt ook om een aantal verbeteringen aan snijschema's te beperken en te sturen.

Al deze hoofdstukken laten kwantum-klassieke scheidingen zien of verschaffen informatieve resultaten voor degenen die een dergelijk verband proberen aan te tonen.

Curriculum Vitæ

From 2016 to 2020 Simon Marshall attended the University of Warwick for an integrated Masters in Maths and Physics. His thesis dissertation concerned the dynamics of cooling metal droplets under James Sprittles in the Maths department. He also completed undergraduate research on quantum collapse models under George Knee and Animesh Datta.

In 2020 he began his PhD in the applied quantum algorithms group in Leiden, under Vedran Dunjko. Initially, his research concerned quantum machine learning and circuit-cutting techniques, but toward the end focused on more complexity theoretical topics. During his PhD studies, he took courses in machine learning for quantum experiments, scientific integrity and scientific conduct, among others.

List of publications

This thesis is based on the following papers:

[141] S. C. Marshall, S. Aaronson & V. Dunjko Improved separation between quantum and classical computers for sampling and functional tasks, arXiv:2410.20935 (2024).

[142] S. C. Marshall, C. Gyurik & V. Dunjko On Bounded Advice Classes, arXiv:2405.18155 (2024).

[143] S. C. Marshall, C. Gyurik & V. Dunjko, High Dimensional Quantum Machine Learning With Small Quantum Computers, *Quantum*, **7**, 1078, 2023.

[144] S. C. Marshall, J. Tura & V. Dunjko, All this for one qubit? Bounds on local circuit cutting schemes, arXiv:2303.13422, 2023.

[35] S. Jerbi, C. Gyurik, S. C. Marshall, R. Molteni & V. Dunjko, Shadows of quantum machine learning, Nature Communications, 15(1), 5676.

In the course of their PhD, the author has additionally authored the following articles that are not included in this thesis:

[145] S. C. Marshall & J. H. Kirchner, Understanding polysemanticity in neural networks through coding theory, arXiv:2401.17975, 2024.

[146] S. Jerbi, C. Gyurik S. C. Marshall, H. J. Briegel & V. Dunjko, Parametrized quantum policies for reinforcement learning, *Advances in Neural Information Processing Systems*, **34**,28362–28375, 2021.

[147] S. Aaronson, S. Grewal, V. Iyer, S. C. Marshall & R. Ramachandran, $\mathsf{PDQMA} = \mathsf{DQMA} = \mathsf{NEXP}$: QMA With Hidden Variables and Non-collapsing Measurements, arXiv:2403.02543, 2024.