Separating quantum and classical computing: rigorous proof and practical application

Marshall, S.C.

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

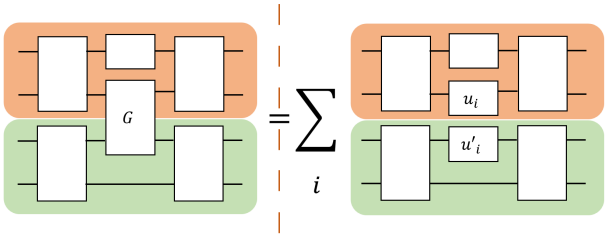


Figure 4.1: The fundamental notion of circuit partitioning. A potential partition (peach coloured and mint coloured) exists but is joined by a 2-qubit gate, G. By expressing G as a sum of single qubit unitaries u_i and u'_i we can simulate the ouput of the large circuit by only running circuits on either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

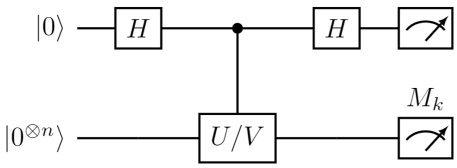


Figure 4.2: The hadamard test to be used for calculating the real component of some $\langle 0| UM_kV |0\rangle$. The controlled U/V circuit implements U if the control is 0 and V if the control is 1. It is important to note that the controlled U/V circuit only requires on control on a few gates, since 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.

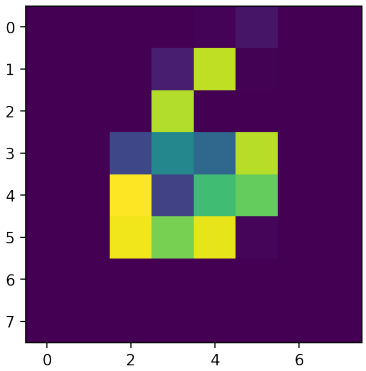


Figure 4.3: An example datum of the handwriting task, the number 6. The picture is then cut into 8 elements (as given by its columns) with each element as input to a different PQC. This resolution data was chosen so as to be fine enough to be human readable.

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.

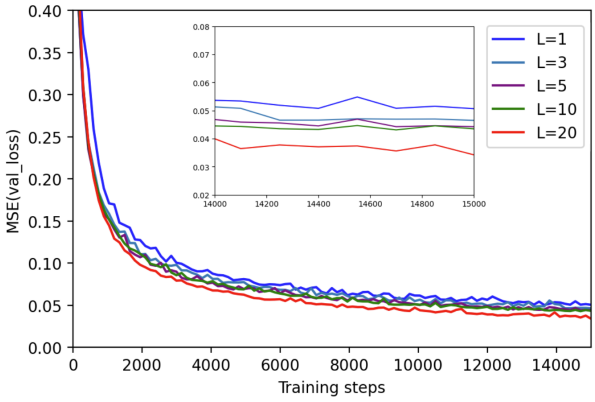


Figure 4.4: The loss on an unseen data set evaluated alongside the training of a reduced partition model to recognise handwritten digits. Increasing the number of terms has a positive effect on the ability of the model.

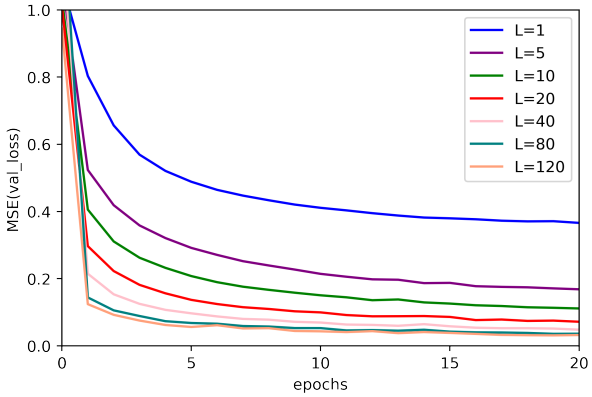


Figure 4.5: The performance of our model on replicating the output of a un-partitioned VQC when θ is fixed to that of the un-partitioned model, and only the parameters meant to replicate the partitioning process are trained. The amount of terms included is varied according to the Legend in the top right.

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.

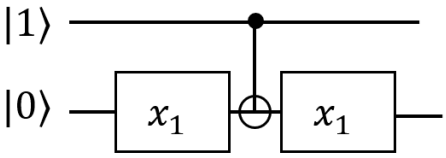
Further work is needed in establishing how many terms (L) might be required for any given task, and what factors influence this requirement. Future work could also focus around the application of this model, testing

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.

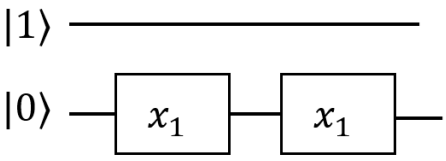
4.A Example of increased complexity after the removal of a 2-qubit gate

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 $\bar{f}_{\boldsymbol{\theta},\boldsymbol{\zeta},\boldsymbol{\lambda}}(x) = \tilde{f}_{\boldsymbol{\theta}}(x) \forall x$.

□