Horizons of Artificial Intelligence in Quantum Computation

by

Bobak T. Kiani

B.S., Johns Hopkins University (2015)
S.M., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

© 2023 Bobak T. Kiani. The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by : Bobak T. Kiani
Department of Electrical Engineering and Computer Science
May 18, 2023

Certified by : Seth Lloyd
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by : Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Horizons of Artificial Intelligence in Quantum Computation

by

Bobak T. Kiani

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The potential emergence of practical quantum computers has guided research into their potential applications, particularly in the context of artificial intelligence. Motivated by the success of deep neural networks in classical machine learning, a prevailing hope is that such success will translate to so-called quantum variational algorithms or quantum neural networks inspired by their classical counterparts.

Contemporary deep learning algorithms are primarily developed using a series of heuristics, which often lack rigorous proofs to justify their efficacy. Due to the opaque nature of these algorithms, providing definitive assurances regarding their performance remains a formidable challenge. Though this complexity extends to the quantum analogues of deep learning, a growing body of literature has identified a set of theoretical tools to better understand the reasons why classical machine learning models are so effective in real-world tasks. We use these tools to investigate these quantum analogues in an effort to partially address the question of when and under what conditions we can anticipate success.

We primarily study the learnability of quantum machine learning algorithms via tools from statistical learning theory, quantum mechanics, random matrix theory, and group theory. Our findings indicate that careful consideration must be given to the design of quantum machine learning algorithms in order to achieve reasonable levels of success. In fact, some of our results reveal that random or unstructured methods in quantum machine learning are prone to various challenges, including issues related to trainability or the absence of significant advantages over the best classical algorithms. Throughout the thesis, we offer several examples of how to potentially introduce structure into these algorithms to partly remedy these issues.

Furthermore, we explore the reverse question of how quantum computing can inform and enhance classical machine learning. We investigate the incorporation of unitary matrices into classical neural networks, which leads to a more efficient design for these unitary neural networks.

Thesis Supervisor: Seth Lloyd
Title: Professor of Mechanical Engineering

Acknowledgments

My experience as a PhD student has been one of many emotions. Getting through the highs and lows of this process would not have been possible without my friends, family, mentors, and colleagues. Any success I have had in this PhD has been because of this wonderful network of people. The list goes on and on, but I hope this acknowledgment section here covers as many as possible.

First and foremost, I want to thank my family for their unwavering support. My mom and dad have been my biggest supporters since childhood, and have encouraged and guided me throughout this process. Mehrdad, thank you for being a wonderful brother and showing me how to be a productive graduate student.

I'm also grateful for the love and support of my extended family, including all my relatives in America and Iran. Special thanks to my grandparents, uncles, and aunts. And additionally to my younger cousins Nima, Ava, Borna, Kasra, Mana, Parsa, and Niki. I appreciate each of you more than words can express. To my girlfriend Sophie, your encouragement and presence in my life have made pursuing a PhD so much more enjoyable and fulfilling. I am so glad you are in my life.

I am greatly thankful for my advisor, Seth Lloyd, for mentoring me and guiding me during the course of my PhD. Your support, guidance, and passion for science have not only enriched my research but also opened up new opportunities for me. To the members of my research group, I am so grateful to have gotten to know you all and work with you over the years. Specifically, I want to thank Milad, David, and Giacomo, all post-docs in our group who have been so helpful and supportive. To my fellow graduate students, Reevu, Ryuji, Can, Lara, Samuel, Andi, and Agnes: thanks for putting up with me and being there at various parts of this journey. This work simply would not be possible without you. To Quynh T. Nguyen, thank you for jumping into our group as a UROP and being a brilliant colleague. And Grecia, thank you as well for being such a wonderful UROP and a great researcher.

I've been fortunate to receive guidance from various faculty members during my PhD studies. Thank you Yann LeCun for your guidance during my internship at Meta and for working with Seth and me on various projects. You have directed me to many interesting avenues of research especially related to self-supervised learning. David Gamarnik, I'm

grateful for the opportunity to work with you and learn from your brilliant research on the probabilistic method and the limits of efficient algorithms. Additionally, I would like to express my gratitude to Dirk Englund, Isaac Chuang, and Aram Harrow for their advice and support at various points in my studies. Finally, thank you to Melanie Weber for taking a chance on me as a post-doctoral student. I am excited to tackle new research questions together.

There are numerous other graduate students and colleagues who I am thankful for. First, I would like to thank Eric Anschuetz for working with me on some of the most challenging problems I have worked on during my PhD. Our work together would not have been possible without your brilliance. Outside of the quantum community, I want to thank many people who have worked with me and helped me learn about interesting problems in computer science. To Hannah Lawrence, thank you for working with me on equivariant machine learning projects and sharing your wonderful ideas. There are a host of people at Meta who I must thank for working with me both during and after my internship over the summer. Thank you Randall Balestriero for helping to invite me to Meta and working with me while I was there. Thank you Vivien Cabannes, Alberto Bietti, Gregoire Mialon, and Quentin Garrido for also working with me on projects at Meta. And finally, to Haggai, Yaron, Derek, and Omri; it was wonderful working with you all at Meta and I hope we keep working together in the future.

I am also thankful to have such wonderful friends. I hope you are all aware of how your presence has made my PhD life so much more fun. Thank you all for being part of this chapter of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning seeks to extract patterns and insights from data. While there are various types of algorithms available, recent progress in the field has focused on the paradigm of deep learning which largely involves a set of fundamental techniques that construct deep networks and train them using gradient based optimization. Despite the simplicity of this approach, it has proven to be remarkably effective in achieving state-of-the-art results in a variety of tasks. The reasons why this paradigm has been so successful are in many ways still unknown.

Alongside the advances and development of deep learning models, quantum machine learning has grown as a field offering quantum analogues to the classical algorithms used in machine learning. Given the remarkable success of these classical models, it is natural to wonder if the same success will translate to the quantum setting. Recent years have seen a growing interest in developing quantum machine learning algorithms and exploring their potential advantages over classical models. Rigorously showing that such advantages exist in the quantum setting has been a formidable challenge.

In this thesis, we explore the question of whether or not quantum models of deep learning can perform as well as the classical models which inspire them. To answer this question in full, we are limited by the fact that we cannot actually run the proposed quantum machine learning algorithms on powerful quantum hardware as such quantum computers have yet to be built. Therefore, the findings in this thesis are mainly devoted to careful theoretical analysis backed in part by numerical experiments. A central theme of this thesis is that care must be taken in the construction of quantum machine learning algorithms for them to

be successful. Directly translating classical techniques into the quantum paradigm without taking into account the unique characteristics of quantum information processing is generally unsuccessful. Instead, successful approaches require leveraging specific structural features of the problem to be practically useful in a wide range of quantum settings.

The main findings are presented in a series of four papers that each cover a different aspect of the overaching goal. Overall, these papers address the topics of

- Showing that common quantum loss landscapes are challenging to optimize:
  [18] Eric R Anschuetz and Bobak T Kiani. Quantum variational algorithms are swamped with traps. *Nature Communications*, 13(1):7760, 2022

- Optimizing a Wasserstein distance metric used to avoid common pitfalls in learning quantum data:
  [205] Bobak Toussi Kiani, Giacomo De Palma, Milad Marvian, Zi-Wen Liu, and Seth Lloyd. Learning quantum data with the quantum earth mover's distance. *Quantum Science and Technology*, 7(4):045002, 2022

- Showing that symmetric quantum systems can be learnable even classically:
  [15] Eric R Anschuetz, Andreas Bauer, Bobak T Kiani, and Seth Lloyd. Efficient classical algorithms for simulating symmetric quantum systems. *arXiv preprint arXiv:2211.16998*, 2022

- Implementing unitary matrices in classical neural networks inspired by approaches in quantum computation:
  [199] Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. projUNN: efficient method for training deep networks with unitary matrices. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022

We address each of these topics in order starting from Chapter 3. First, in this introduction section, we provide some mathematical background and key context for the later chapters.

In addition to the works above, there are a mixed set of research topics that were part of my graduate studies, but not included in this thesis. For sake of completeness, they are grouped into categories and briefly described below.

- **Equivariance and symmetries in deep networks:** Symmetries play an important role in the design of neural networks for machine learning [221]. Aiming to quantify or theoretically understand the benefits of incorporating such symmetries and equivariance properties into neural networks, we characterized the implicit bias of a class of equivariant networks during training [219]. A popular class of equivariant neural networks are graph neural networks, those used on input data which is in the form of a graph. For these networks, we characterized the equivariant graph polynomials and enhanced graph neural networks with features from these polynomials [279].

- **Quantum algorithms:** As a member of Seth Lloyd's lab group, a number of studies were made into the design of quantum algorithms, mostly led by other members of the lab group. These studies resulted in quantum algorithms for various tasks including solving differential equations [236, 202], implementing structured matrices [259, 73, 207], and performing the polar decomposition of a matrix [236].

- **Hamiltonian complexity:** Late into my PhD studies, I became interested in studying the complexity of learning Hamiltonians which guided me into topics in Hamiltonian complexity. With David Gamarnik and Eric Anschuetz, we designed a proof of the combinatorial no low-energy trivial state (cNLTS) theorem for random Hamiltonians motivated by the overlap gap property [17].

- **Theoretical analysis of deep networks and self-supervised learning:** Understanding why and how neural networks learn so well is an open topic of research in machine learning [351]. With members of my lab group and at Meta AI research, we conducted two different studies in this direction. The first direction studied random neural networks, showing that they are biased towards simple functions [106] and relatively robust to small input perturbations [107]. The second direction analyzed self-supervised learning, which is a popular unsupervised learning framework, and proved various theorems on the performance of self-supervised learning in the kernel regime [61].

1.1 Quantum Mechanics Background

Quantum mechanics is a fundamental theory of physics that studies physical phenomena at the atomic scale. Here, we briefly review the mathematical axioms and basic concepts of quantum mechanics that are applied throughout this work. A detailed and pedagogic background to the concepts and ideas can be found in many reference textbooks [291, 117, 326, 149].

We briefly overview the central mathematical concepts of quantum mechanics by sharing four axioms which govern the form of quantum states, their evolution, and the measurements used to extract information from them. These axioms come from the Dirac-von-Neumann formulation of quantum mechanics [117, 326]. Throughout this thesis, we will only consider finite dimensional Hilbert spaces and simplify these axioms to, in some cases, only hold for this specific setting. Therefore, caution must be taken if at any point, the ideas here are attempted to be abstracted into the infinite dimensional setting. Finally, we use the term axiom here slightly loosely as one could argue that certain of these axioms are in fact properties that can be derived from the others.

Axiom 1.1.1. *The space of quantum states forms a separable complex Hilbert space $\mathcal{H}$ with inner product $\langle\cdot|\cdot\rangle$. A state is any positive trace-class linear map $\rho : \mathcal{H} \to \mathcal{H}$ for which $\mathrm{Tr}(\rho) = 1$. For finite dimensional Hilbert spaces, a state is pure if $\mathrm{Tr}(\rho\rho) = 1$. Such pure states can be associated to a unit-norm $|\psi\rangle \in \mathcal{H}$ where $\rho\,|\phi\rangle = \langle\psi|\phi\rangle\,|\psi\rangle$ for all unit-norm $|\psi\rangle \in \mathcal{H}$.*

Throughout this text, we use bra-ket notation to denote quantum states and other operations. A ket $|\psi\rangle \in \mathcal{H}$ denotes a quantum pure state (element of the Hilbert space) whereas a bra $\langle\psi|$ denotes the conjugate transpose of $|\psi\rangle$ (again, assuming that we are in a finite dimensional Hilbert space).

Axiom 1.1.2. *The observables of a quantum system are the self-adjoint linear operators $\boldsymbol{A} : \mathcal{H} \to \mathcal{H}$ corresponding to Hermitian matrices in finite dimensions.*

In other words, observable matrices correspond to those matrices $\boldsymbol{A}$ for which $\boldsymbol{A}^\dagger = \boldsymbol{A}$.

Axiom 1.1.3. *The expectation of an observable $\boldsymbol{A}$ with respect to a state ρ is equal to $\mathrm{Tr}(\boldsymbol{A}\rho)$. For pure states $|\psi\rangle$, this is equivalent to $\langle\psi|\,\boldsymbol{A}\,|\psi\rangle$.*

Axiom 1.1.4. *Evolution of a quantum pure state is governed by the Schrödinger equation*

$$i\hbar\frac{d}{dt}|\Psi(t)\rangle = \boldsymbol{H}|\Psi(t). \tag{1.1}$$

In many cases, the constant $\hbar$ is subsumed within the Hamiltonian $\mathcal{H}$.

Evolution with a fixed Hamiltonian $\boldsymbol{H}$ that does not vary over time results in unitary evolution $|\Psi(T)\rangle = \exp(-i\boldsymbol{H}T/\hbar)\,|\Psi(0\rangle$, where $\exp(-i\boldsymbol{H}T/\hbar)$ is a unitary operator.

1.1.1 Quantum Computation

Quantum computers apply the formalism of quantum mechanics to computing tasks. A fundamental building block of quantum computation is the qubit, a two-level quantum system that can be in a superposition of both states $|0\rangle$ and $|1\rangle$ at the same time. Here $|0\rangle$ and $|1\rangle$ are linear basis elements of the complex Hilbert space $\mathbb{C}^2$. Any single qubit state $|\psi\rangle$ by a complex linear combination

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.2}$$

where $|\alpha|^2$ and $|\beta|^2$ give the probabilities of measuring the qubit in states $|0\rangle$ and $|1\rangle$ respectively, and $|\alpha|^2 + |\beta|^2 = 1$.

Higher dimensional building blocks of dimension d are labeled qudits with a basis denoted as $|0\rangle, |1\rangle, ..., |d-1\rangle$. Similarly, the state of a qudit can be described by a complex linear combination of its basis states:

$$|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i|i\rangle,$$

where $|\alpha_i|^2$ gives the probability of measuring the qudit in the state $|i\rangle$.

Computation on a quantum state can be represented in the quantum circuit formalism. When two or more qubits are combined, the resulting state is given by the tensor product of the individual states. For example, given two states $|\psi_1\rangle \in \mathbb{C}^{d_1}$ and $|\psi_2\rangle \in \mathbb{C}^{d_1}$ placed in a quantum circuit as separate "registers", the combined state is given by $|\psi_1\rangle \otimes |\psi_2\rangle \in \mathbb{C}^{d_1 d_2}$.

In the quantum circuit formalism, computation is peformed by applying quantum gates to subsets of qubits. Each quantum gate is a unitary operator acting on one or more qubits. As in classical computation, two qubit gates are sufficient to perform any arbitrary unitary on a larger subsystem of n qubits. Each quantum gate can be parameterized by a Hamiltonian,

which is a Hermitian operator that generates the gate via time evolution. The evolution of a quantum system under a Hamiltonian H is governed by the Schrödinger equation:

$$i\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle, \tag{1.3}$$

where $|\psi(t)\rangle$ is the state of the system at time t, and H is the Hamiltonian operator. The constant $\hbar$ typically shown in the Schrödinger equation is implicitly assumed to be contained within the Hamiltonian normalization.

Some commonly used quantum gates in quantum computation are the Pauli gates, the Hadamard gate, the phase gate and the CNOT gate. The Pauli correspond to the three Pauli matrices:

$$\boldsymbol{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.4}$$

The Hadamard gate is defined as:

$$\boldsymbol{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.5}$$

The phase gate is defined as:

$$\boldsymbol{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{1.6}$$

The CNOT gate is defined as:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.7}$$

Quantum computation can be visually shown as a quantum circuit analogous to those in classical computation. An example is shown in Figure 1-1 which constructs the GHZ state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This circuit initializes the quantum state as $|00\rangle$ and applies a Hadamard gate followed by a CNOT gate.

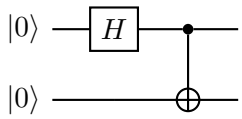


Figure 1-1: Example circuit constructing the 2 qubit GHZ state.

1.2 Unitary and Orthogonal groups

Lie groups, and specifically unitary and orthogonal matrices, play a significant role in the fields of physics and machine learning. In the context of machine learning, Lie groups often arise when dealing with transformations and symmetries. For example, they can be used to model rotations and translations in computer vision or to design efficient algorithms for optimization on manifolds. Furthermore, unitary and orthogonal matrices are particularly relevant due to their desirable properties, such as preserving the inner product and maintaining stability in iterative processes.

Quantum mechanics, which forms the foundation of quantum computation, is inherently tied to unitary matrices which represent the evolution of these systems over time. Here, we provide a brief mathematical background to Lie groups and Lie algebras with a particular focus on the unitary and orthogonal groups. For a more comprehensive overview, we recommend [159].

Though Lie groups are typically defined with respect to differentiable manifolds, we restrict ourselves here to the subset of matrix Lie groups which is less general but allows for a more concise and simpler theoretical overview. Informally, Lie groups are groups whose elements are specified by a set of parameters that vary smoothly and continuously, *i.e.,* the group is also a differentiable manifold. Specific to matrices, Lie groups are commonly defined with respect to the general linear group $GL(n, \mathbb{C})$ denoting the set of invertible $n \times n$ matrices with complex valued entries [159]. Lie groups are closed subgroups of $GL(n, \mathbb{C})$ that have the following smoothness property.

Definition 1.2.1 (Matrix Lie groups [159])**.** A matrix Lie group is any subgroup of $GL(n, \mathbb{C})$ with the property that any sequence of matrices $\boldsymbol{A}_m \in \mathbb{C}^{n \times n}$ in the subgroup converge to a matrix $\boldsymbol{A}$ that is either an element of the subgroup or not invertible (*i.e.,* not in $GL(n, \mathbb{C})$).

Two important Lie groups studied in this work are the unitary and orthogonal groups

whose definitions are copied below.

$$O(n) = \left\{ \boldsymbol{M} \in \mathbb{R}^{n\times n} | \boldsymbol{M}\boldsymbol{M}^\intercal = \boldsymbol{M}^\intercal\boldsymbol{M} = \boldsymbol{I} \right\}, \tag{1.8}$$

$$U(n) = \left\{ \boldsymbol{M} \in \mathbb{C}^{n\times n} | \boldsymbol{M}\boldsymbol{M}^\dagger = \boldsymbol{M}^\dagger\boldsymbol{M} = \boldsymbol{I} \right\}. \tag{1.9}$$

The Lie algebra is the tangent space of a Lie group at the identity element. To observe this, we introduce the matrix exponential map which is central to the connection between Lie groups and their corresponding Lie algebras.

$$\exp(\boldsymbol{M}) = \sum_{k=0}^{\infty} \frac{\boldsymbol{M}^k}{k!}. \tag{1.10}$$

For compact groups, the exponential map is a smooth map whose image is the connected component to the identity of the Lie group [213, 159]. The special orthogonal and unitary groups are both compact and connected so the exponential map is surjective for these groups (*i.e.,* for every group element, there exists an element of the Lie algebra whose exponential is equal to that group element). However, the orthogonal group has two connected components, *i.e.,* elements with positive and negative determinant, and the image of exponential map are only orthogonal matrices with positive determinant.

Since the matrix exponential map is a smooth function, we can take the derivative of the exponential map with respect to a parameter as below.

$$\frac{d}{dt}\exp(t\boldsymbol{X}) = \boldsymbol{X}\exp(t\boldsymbol{X}) = \exp(t\boldsymbol{X})\boldsymbol{X}, \tag{1.11}$$

and thus,

$$\frac{d}{dt}\exp(t\boldsymbol{X})\Big|_{t=0} = \boldsymbol{X}. \tag{1.12}$$

The above gives us the Lie algebra to a given group.

Definition 1.2.2 (Lie algebra [159])**.** Given a Lie group G, the Lie algebra $\mathfrak{g}$ of G is the set of matrices $\boldsymbol{X}$ such that $e^{t\boldsymbol{X}} \in G$ for all $t \in \mathbb{R}$.

Typically, Lie algebras of a Lie group are denoted with Gothic or Fraktur font. Using the above definition, one can construct the corresponding Lie algebras. As an example, consider

the unitary group where given a matrix $\boldsymbol{U} \in U(n)$ and $\boldsymbol{X} \in \mathfrak{u}(n)$,

$$\boldsymbol{U}^{-1} = \boldsymbol{U}^{\dagger} \iff \exp(t\boldsymbol{X})^{-1} = \exp(-t\boldsymbol{X}) = \exp(t\boldsymbol{X})^{\dagger} = \exp(t\boldsymbol{X}^{\dagger}), \tag{1.13}$$

and since the above holds for all $t \in \mathbb{R}$, we can differentiate the above at $t = 0$, obtaining the property of elements of the unitary Lie algebra that $-\boldsymbol{X} = \boldsymbol{X}^{\dagger}$ as seen in the main text:

$$\begin{aligned} \frac{d}{dt} \exp(-t\boldsymbol{X})\Big|_{t=0} &= \frac{d}{dt} \exp(t\boldsymbol{X}^{\dagger})\Big|_{t=0} \\ -\boldsymbol{X} &= \boldsymbol{X}^{\dagger}. \end{aligned} \tag{1.14}$$

Proceeding in a similar fashion with the orthogonal group, we have that

$$\mathfrak{o}(n) = \left\{ \boldsymbol{A} \in \mathbb{R}^{n \times n} : \boldsymbol{A} + \boldsymbol{A}^{\intercal} = 0 \right\}, \tag{1.15}$$

$$\mathfrak{u}(n) = \left\{ \boldsymbol{A} \in \mathbb{C}^{n \times n} : \boldsymbol{A} + \boldsymbol{A}^{\dagger} = 0 \right\}. \tag{1.16}$$

1.3 Machine Learning

Machine learning problems generally consist of a given input dataset from which to learn and an algorithm which is applied to that dataset to perform learning. Constructing algorithms that can fit models to a given dataset is in general a challenging task. The ultimate goal of machine learning is to use such models to extract meaningful patterns and make accurate predictions on new, unseen data.

The modern paradigm of deep learning is based on the empirical risk minimization (ERM) framework, which formalizes the learning problem as an optimization task, aiming to minimize the empirical risk, which is the average loss incurred by a model over the training dataset. For example, in the supervised learning framework, one may be given a dataset of N input-output pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Given a parameterized function $f_\theta : \mathcal{X} \to \mathcal{Y}$ mapping inputs to outputs where its parameters θ are tunable and a loss function $\ell(\cdot, \cdot)$ where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, the optimization task in ERM aims to minimize the empirical loss

$$\widehat{\mathcal{R}} = \sum_{i=1}^{N} \ell\left(f_\theta(\boldsymbol{x}_i), y_i\right). \tag{1.17}$$

For example, in a regression task, the inputs $\boldsymbol{x}_i \in \mathbb{R}^d$ may be vectors and the targets

$y_i \in \mathbb{R}$ may be numbers which we aim to regress by minimizing the mean squared error loss $\ell\left(f_\theta(\boldsymbol{x}_i), y_i\right) = \left(f_\theta(\boldsymbol{x}_i) - y_i\right)^2$.

The loss function quantifies the discrepancy between the model's predictions and the true target values, while the empirical risk represents the expectation of this loss over the training data. By minimizing the empirical risk, the learning algorithm is effectively finding the best possible model parameters to fit the data and consequently achieve good generalization performance on unseen data points.

Deep neural networks are a method of parameterizing trainable functions f_θ via multiple layers of carefully designed transformations [221]. These networks are generally capable of learning complex, hierarchical representations of the input data and achieve state-of-the-art performance on a wide range of tasks, including image recognition, natural language processing, and game playing [59, 302].

The training of deep networks is typically done via gradient-based optimization methods, such as stochastic gradient descent (SGD) and its variants [221, 212]. These methods iteratively update the model's parameters by following a heuristic which tracks in the direction of the negative gradient of the loss function with respect to the parameters (e.g., $-\nabla_\theta \widehat{\mathcal{R}}$ for our example in Equation (1.17)). This process is performed iteratively, with each update taken over a random batch of training data, until convergence to a satisfactory solution is achieved or a predetermined stopping criterion is met.

Although deep learning has achieved remarkable success across a broad range of applications, generalization remains a critical challenge in the field. Generalization refers to a model's ability to perform well on new, unseen data points, and is central to the utility of machine learning models in practical settings. The capacity of deep networks to fit complex patterns can sometimes lead to overfitting, where the model learns the noise in the training data rather than the underlying structure, resulting in poor generalization performance.

The understanding of generalization in deep learning is still incomplete, and the development of a comprehensive theoretical framework for understanding and improving generalization in deep networks remains an open research challenge [351]. As we explore the potential of quantum computation to revolutionize the field of machine learning, understanding and addressing the generalization challenge in the context of quantum machine learning is essential for the successful development and deployment of quantum-enhanced learning algorithms.

1.3.1 Quantum Machine Learning

A central area of interest within quantum machine learning is the development of quantum variational algorithms or quantum neural networks (QNNs) [76, 50, 295]. QNNs are quantum analogs of classical neural networks, in which the trainable parameters are replaced by parameterized quantum operations and input data take the form of quantum states. These networks aim to harness the power of quantum computation to enable more efficient and powerful learning algorithms, capable of solving complex problems that are currently intractable for classical deep learning methods.

While the impressive generalization properties of classical deep learning models have been widely observed, it is important not to assume that these properties will automatically translate to their quantum counterparts. Simply providing a learning algorithm with increased expressibility does not directly translate to performance gains as our short discussion of generalization points out. Quantum states reside in a much larger Hilbert space than classical states, which implies an exponentially larger state space for quantum systems compared to their classical counterparts. This larger state space offers the potential for more expressive QNNs, which could, in principle, lead to improved learning capabilities. However, the larger Hilbert space also presents challenges in terms of optimization and generalization, as the increased complexity can lead to a more difficult optimization landscape.

As we will see, handling this exponential space will be one of the challenges we look at closely in this thesis. Other challenges may also arise including issues with state preparation, measurement noise, and the probabilistic nature of quantum mechanics. Each of these requires a careful analysis of the quantum machine learning which we study in these pages.

Chapter 2

Traps in Variational Algorithms

2.1 Introduction

The trainability of classical neural networks via simple gradient-based methods is one of the most important factors leading to their general success on a wide variety of problems. This is particularly exciting given the variety of no-go results via statistical learning theory, which demonstrate that in the worst case these models are not trainable via stochastic gradient-based methods [53, 312, 140, 297]. There has been recent hope that variational quantum algorithms—the quantum analogue of traditional neural networks—may inherit these nice trainability properties from classical neural networks. Indeed, in certain regimes [129], training algorithms exist such that the resulting quantum model provably outperforms certain classical algorithms. This would potentially enable the use of quantum models to efficiently represent complex distributions which are provably inefficient to express using classical networks [137].

Unfortunately, such good training behavior is not always the case in quantum models. There have been previous untrainability results for deep variational quantum algorithms due to vanishing gradients [246, 74, 242, 256], and for nonlocal models due to poor local minima [20]; however, no such results were known for shallow, local quantum models with local cost functions. Indeed, there have been promising preliminary numerical experiments on the performance of variational quantum algorithms in these regimes, but typically have relied on good initialization [97] or highly symmetric problem settings [335, 210, 211] to show convergence to a good approximation of the global optimum.

Here, we show that generally such models are not trainable, particularly when a good

choice of initial point is not known and when the model does not exhibit a high amount of symmetry. We first prove general untrainability results in the presence of noise using techniques from statistical query learning theory. Surprisingly, these results hold for all learning problems in a wide range of variational learning settings, and in many scenarios even when the magnitude of the noise is exponentially small in the problem size. We then consider the trainability of models that may not have noise by studying their typical loss landscapes. We prove that, for typical model instances, local minima concentrate far from the global optimum even for certain local shallow circuits that do not suffer from barren plateaus. This phenomenon can be visualized in Fig. 2-1, where the training landscape for a shallow QCNN learning a random instance of itself is shown to concentrate far from the global optimum. As in [20], this phenomenon is the result of a trainability phase transition in the loss landscape of the quantum model. In [20], this transition was governed by the ratio of the number of parameters to the Hilbert space dimension; we show in the shallow case that instead, this transition is governed by the ratio of the local number of parameters to the local Hilbert space dimension, in the reverse light cone of a given measured observable. As this is typically much less than 1 for local variational ansatzes, these models are typically untrainable. We then give numerical evidence of this fact, and conclude by studying where there may be reason for optimism in the training of certain variational quantum models.

2.2 Results

2.2.1 Preliminaries

Quantum machine learning algorithms have been a focus of intense research effort as potential use-cases for noisy, intermediate-scale quantum (NISQ) [278] devices. Just as in classical machine learning, algorithms are tasked with minimizing some risk:

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}} \left[\ell(f(\boldsymbol{x})) \right], \tag{2.1}$$

given a model f, a distribution of inputs $\mathbf{x}$, and a loss function ℓ. To perform learning, one searches for a model $\widehat{f} \in \mathcal{F}$ in the function class $\mathcal{F}$ (e.g. the set of functions expressed by quantum neural networks). The expected risk $\mathcal{R}(\widehat{f})$ is typically not something one can calculate, as it requires access to the full probability distribution of the data. Instead,

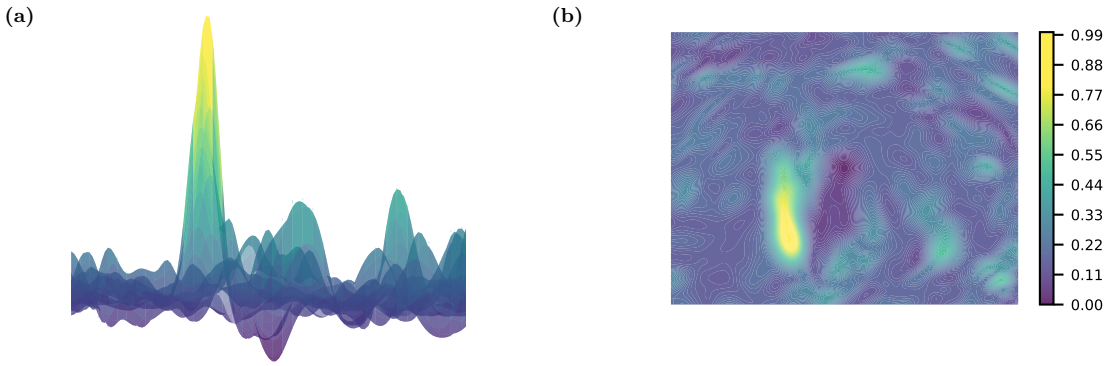


Figure 2-1: **Typical shape of loss landscape.** Loss landscapes of underparameterized quantum variational algorithms generally appear "bumpy," filled with various local minima and traps. Here, we plot the loss landscape as a (a) surface and (b) contour plot along two random normalized directions for the teacher-student learning task of the QCNN for 14 qubits. Though a global minimum is located at the center of the plot, finding this global minima is generally challenging due to the shape of the loss landscape. Details of this visualization are given in Supplementary Note 6.

one often minimizes the empirical risk $\widehat{\mathcal{R}}(\widehat{f})$ (often named the training error) over a given training data set D:

$$\widehat{\mathcal{R}}(\widehat{f}) = \sum_{\boldsymbol{x}_i \in D} \ell(\widehat{f}(\boldsymbol{x}_i)). \tag{2.2}$$

Perhaps the most well-studied class of quantum machine learning algorithm consists of variational quantum algorithms (VQAs) [273]. VQAs are a class of quantum generative models where one expresses the solution of some problem as the smallest eigenvalue and corresponding eigenvector (typically called the ground state) of an objective Hermitian matrix $\mathbf{H}$—called the Hamiltonian—on n qubits. Given a choice of generative model—often called an ansatz in the quantum algorithms literature:

$$|\boldsymbol{\theta}\rangle = \prod_{i=1}^{q} \mathbf{U_i}\left(\theta_i\right)|\psi_0\rangle \tag{2.3}$$

that for some choice of $\boldsymbol{\theta}$ is the ground state of $\mathbf{H}$, the solution is encoded as the minimum of the loss function

$$\widehat{\mathcal{R}}_{\mathrm{VQE}}(\boldsymbol{\theta}) = \sum_{i=1}^{A} \alpha_i \langle\boldsymbol{\theta}|\,\mathbf{P}_i\,|\boldsymbol{\theta}\rangle\,, \tag{2.4}$$

where:

$$\mathbf{H} = \sum_{i=1}^{A} \alpha_i \mathbf{P}_i \tag{2.5}$$

is the Pauli decomposition of $\mathbf{H}$. VQAs have found numerous applications [76], and a countless number of VQA instances have been proposed for various quantum learning tasks.

Typically, models in VQAs come in one of two flavors: Hamiltonian agnostic models, and Hamiltonian informed models. Hamiltonian agnostic models are constructed such that the $\mathbf{U_i}$ are independent of $\mathbf{H}$, and are generally chosen to be efficient to implement. This is most analogous to the case in classical generative modeling, where the model structure is usually independent from the specific choice of data $\mathbf{H}$. One might hope then that training Hamiltonian agnostic VQAs is completely analogous to the classical setting, then, and the loss landscape of equation 2.4 exhibits the desirable properties that enable trainability found in classical networks [92, 82].

Unfortunately, unlike the classical setting, the performance of VQAs is often dominated by poor performance in the training procedure (see Supplementary Note 1 for a discussion). For one, VQAs tend to exhibit barren plateaus when they are deep; namely, gradients of deep variational quantum circuits vanish exponentially with the problem size in many settings [246, 74, 256]. Problematic training in this regime has also been studied beyond gradient descent [77, 28].

Until recently, less was known about the trainability of VQAs in the shallow model regime. Numerically, [206, 335] showed that randomly chosen variational landscapes typically have poor local minima, a result which was later proven in [20] for nonlocal models using tools from random matrix theory. In a similar line of research, [345] showed that for certain quantum variational ansatzes or quantum neural networks, there exist data sets and loss functions which induce exponentially many local minima in the loss landscape. [232, 343] both showed that, in an overparameterized regime, these models experience good local minima, though this transition to trainability typically occurs at an intractable number of parameters. Finally, assuming the presence of a constant rate of noise per ansatz gate, [112] showed convergence of the loss landscape to the uniform distribution at a rate exponential in the circuit depth. Many of these previous results on the untrainability of VQAs are summarized in Table 2.2, along with a summary of our results which focus on the shallow, local regime.

2.2.2 Learning in the Statistical Query Framework

Quantum machine learning algorithms are inherently noisy due to both unavoidable sources of error—such as shot noise from sampling outputs—or potentially correctable sources of error such as gate errors and state preparation noise. In such noisy settings, the statistical query (SQ) model provides a useful framework for quantifying the complexity of learning a class of functions by considering how many query calls to a noisy oracle are needed to learn any function in that class (see Supplementary Note 2 for a brief review and history of SQ models) [139, 194, 312]. In this setting, we consider the optimization of a risk of the form of equation 2.2. We assume there is a target observable $\boldsymbol{M}$ that we would like to learn on some distribution over states $\mathcal{D}$. We define a correlational statistical query $\mathrm{qCSQ}(\boldsymbol{O}, \tau)$, which takes in a bounded observable $\boldsymbol{O}$ with $\|\boldsymbol{O}\| \leq 1$ and a tolerance τ and returns a value in the range:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho) - \tau\right] \leq \mathrm{qCSQ}(\boldsymbol{O}, \tau) \leq \mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho) + \tau\right]. \tag{2.6}$$

Note that there are no guarantees on the form of the additive error other than it is within the tolerance τ, and may for instance depend on the observable being queried $\mathbf{O}$. Though SQ oracle calls may at first appear unrelated to variational algorithms, we show in the Methods that many common variational optimizers in the presence of noise of the magnitude τ reduce to calls to an SQ oracle; for instance, commonly used first and second order optimization algorithms fall within the framework of the SQ model we consider. In the Methods, we also describe an analogous SQ model for learning unitaries.

To quantify the hardness of learning variational circuits, we consider the task of learning certain function classes generated by shallow variational circuits over a distribution of inputs $\mathcal{D}$ which forms a 2-design. Our results also generally hold for distributions that are uniform over states in the computational basis, recovering the statistical query setting for classical Boolean functions. Table 2.1 summarizes the number of queries needed to learn various function classes which are generated by variational circuits, with proofs deferred to Supplementary Note 3. In all settings we consider, an exponential number of queries (in either n or the light cone size) are needed to learn simple classes, such as the class of functions generated by single qubit gates followed by a fixed global measurement. This hardness intuitively arises because each individual query can only obtain information about a few

Setting (n qubits, L layers)	Query complexity ($\beta < 1/2^*$)
$L = 1$, global measurement, single qubit gates	$2^{\Omega(n)}$ if $\tau \geq 3^{-\beta n}$
$L = \lceil \log_2 n \rceil$, single qubit measurement, global 1- and 2-local gates	$2^{\Omega(n)}$ if $\tau \geq 4^{-\beta n}$
$L \ll n$, single qubit measurement, neighboring 1- and 2-local gates on a d-dim. lattice	$2^{\Omega(L^d)}$ if $\tau = \Omega(1)^{**}$
$L = 1$, single qubit gates, unitary learning	$2^{\Omega(n)}$ if $\tau \geq 4^{-\beta n}$

* Technically, we require $\beta = 1/2 - \Omega(1)$; ** $\tau = 2^{-\omega\left(\min\left(2L, n^{1/d}\right)^d\right)}$ is sufficient.

Table 2.1: Relatively simple classes of functions require exponentially many statistical queries to learn using any naive algorithm that reduces to statistical queries. The table above quantifies the number of queries needed to identify a target function in the function class, over a distribution of states that forms a 2-design and with queries that have tolerance $C_{max}\tau$ (query tolerance lower bounded by a constant times C_{max} suffices in all cases).

of the exponentially many orthogonal elements in the function class. More formally, we lower bound the SQ dimension (defined in the Methods) of the function classes considered in Table 2.1 to show our query lower bounds.

Our hardness results hold for any target observable $\mathbf{M}$, as long as the learning setting is one we consider in Table 2.1. Furthermore, they hold for any variational ansatz—not just on average—provided it is in one of the settings of Table 2.1. Finally, our results hold for any constant error τ in the statistical queries; indeed, the majority of our results hold even if this noise were only exponentially small in the problem size. For instance, training via gradient descent where the gradient is estimated using polynomially many samples fits into this framework immediately just from the induced shot noise.

In a more positive light, learning local Hamiltonians generated by shallow depth circuits can potentially be efficiently performed as the complexity grows exponentially only with locality or depth in this setting. In fact, prior results have provably shown that certain classes of Hamiltonians are efficiently learnable using properly chosen algorithms [21, 34]. Nevertheless, this does not correspond to efficient learnability of the ground state of a given Hamiltonian, as learnability of the properties of a Hamiltonian is not the same as the learnability of its ground state. Indeed, we will see in Sec. 2.2.3 that typically, even in this setting, learning the ground state of such a local Hamiltonian is difficult.

Our hardness results do not indicate that simple classes of functions like those generated by single qubit rotations are hard to learn for all algorithms, but only those whose steps

reduce to statistical queries. For example, the class of Pauli channels is not learnable in the SQ setting, but there exist simple, carefully constructed, algorithms which can learn Pauli channels [86, 178, 141]. This is analogous to the classical setting where parity functions are hard to learn in the noisy SQ setting, but efficient to learn using simple linear regression [194]. Similarly, the related work of [169] showed that output distributions of Clifford circuits can be hard to learn using statistical queries, but efficient using a technique that resorts to linear regression on a matrix formed from samples of the overall distribution. More loosely, our results provide support to the basic maxim that algorithms which apply too broadly will work very rarely [337]; more careful construction of learning algorithms tailored to the problem at hand is generally necessary. One straightforward way to avoid the hardness of the SQ setting is to construct algorithms whose basic steps do not reduce to statistical queries, e.g. via the construction of non-global metrics [203, 175, 197]. However, such a fix is by no means guaranteed to avoid the more general issues of poor landscapes and noise that also make learning in the SQ setting so difficult, as we now examine.

2.2.3 Loss Landscapes of Local Variational Quantum Algorithms

We now consider the trainability of VQAs in the noise free regime, beyond optimization algorithms that reduce to statistical queries. Though we are unable to prove the very strong no-go results proved in the SQ framework, we are able to show that the loss landscapes of typical local variational algorithms with Hamiltonian agnostic ansatzes are unamenable to optimization. We achieve this by showing that typically, the loss landscapes of shallow, local VQAs are swamped with poor local minima.

As discussed in Table 2.2, it is already known that deep Hamiltonian agnostic ansatzes are typically untrainable due to the presence of barren plateaus [246, 74, 256]; hence, here we focus on shallow ansatzes. Previous results [20] have also shown that shallow, nonlocal models are untrainable, by showing that the scrambling of variational ansatzes over the entire system in these instances induce poor local minima. These techniques were not extendable to shallow, local ansatzes, however, which do not scramble globally.

Instead, here, we show that ansatzes that approximately scramble locally are difficult to train. As we will later show, this includes common classes of variational ansatzes, such as Hamiltonian agnostic checkerboard ansatzes on a d-dimensional lattice. We show that this approximate, local scrambling suffices to imply that the loss landscapes of these VQAs are

Result	Dimension	Locality	Depth	Worst case?	Barren plateaus?	Poor minima?
[246]	d	2	$\Omega\left(n^{\frac{1}{d}}\right)$	✗	✓	?
[74]	1	2	$\omega\left(\log\left(n\right)\right)$	✗	✓	?
[256]	d	2	$\omega\left(\log\left(n\right)^{\frac{1}{d}}\right)$	✗	✓	?
[20]	N/A	n	$\Omega\left(1\right)$	✗	✓/✗	✓/✗
[345]	d	2	$\Omega\left(1\right)$	✓	?	✓
Our results	d	2	$\Omega\left(1\right)$	✗	✗	✓

Table 2.2: A summary of previous results on the untrainability of variational quantum algorithms. A label of "✓/✗" denotes that the paper studied certain regimes where the phenomenon was present, and certain regimes where it was not. A label of "?" denotes that the phenomenon was not studied. "Dimension" indicates the locality structure of the ansatzes study. For instance, Dimension = 1 denotes ansatzes with nearest-neighbor interactions for qubits on a line. "Worst case" denotes analysis performed with adversarial data.

close to those of Wishart hypertoroidal random fields (WHRFs). These are random fields parameterized by l, m of the form:

$$F_{\text{WHRF}}\left(\mathbf{w}\right) = m^{-1} \sum_{i,j=1}^{2^l} w_i J_{i,j} w_j, \tag{2.7}$$

where $\mathbf{J}$ is drawn from a Wishart distribution with m degrees of freedom, and $\mathbf{w}$ are points on a certain embedding of the hypertorus $\left(S^1\right)^{\times l}$ in $\mathbb{R}^{2^l}$. We demonstrate this convergence via new techniques, directly bounding the error in the joint characteristic function of the function value, gradient, and Hessian components of the variational loss from those of WHRFs. As the typical loss landscapes of WHRFs are known given these random variables (see Methods for a summary), by demonstrating sufficient convergence of these random variables to those of WHRFs, we will be able to infer the distribution of critical points for local VQAs.

To begin, we take our (assumed traceless) problem Hamiltonian to have Pauli decomposition:

$$\mathbf{H} = \sum_{i=1}^{A} \alpha_i \mathbf{P_i}, \tag{2.8}$$

and for simplicity scale and shift the loss landscape of equation 2.4 to be of the form:

$$\widehat{\mathcal{R}}_{\text{VQE}}\left(\boldsymbol{\theta}\right) = 1 + \left\|\boldsymbol{\alpha}\right\|_1^{-1} \sum_{i=1}^{A} \alpha_i \left\langle\boldsymbol{\theta}\right| \mathbf{P_i} \left|\boldsymbol{\theta}\right\rangle, \tag{2.9}$$

where $\boldsymbol{\alpha}$ is the vector of all α_i and the ansatz $\left|\boldsymbol{\theta}\right\rangle$ is as given in equation 2.3. As this ansatz is assumed to be shallow and local, we assume that the reverse light cone of each $\mathbf{P_i}$ under

the ansatz is of size $l \ll n$.

As in most analytic treatments of Hamiltonian agnostic VQAs, we consider certain randomized classes of ansatzes [246, 74, 256, 20]. Roughly, we assume that in a local region around each measured Pauli observable $\mathbf{P_i}$, the ansatz is an ϵ-approximate t-design; that is, its first t moments are ϵ-close to those of the Haar distribution. This is a much weaker assumption than global scrambling of the ansatz. For instance, for $\mathbf{P_i}$ of constant weight, such approximately locally scrambling circuits include constant depth local circuits with random local gates [161]. We discuss in more detail when this assumption holds in practice when specializing to common variational quantum learning scenarios, and defer technical details to Supplementary Note 4.

Our main result, informally, is that the random field given by equation 2.9 under this approximate, local scrambling assumption converges in distribution to that of a WHRF. The formal statement and derivation of this result are given in Supplementary Note 4, where we also lay out our assumptions more explicitly. Informally, the result follows by deriving a bound on the error in the joint characteristic function of the loss function and its first two derivatives from that of a WHRF. We then use this to bound the error in distribution that is incurred by the induced scrambling being only approximate. Finally, we show using properties of local Haar random gates and the locality of the problem Hamiltonian that this suffices to prove convergence of these random variables to those of a WHRF.

Theorem 2.2.1 (Approximately locally scrambled variational loss functions converge to WHRFs, informal)**.** *Let*

$$m \equiv \frac{\|\boldsymbol{\alpha}\|_1^2}{\|\boldsymbol{\alpha}\|_2^2} 2^{l-1} \tag{2.10}$$

be the degrees of freedom parameter. Assume $q \log(q) = \mathrm{o}(m)$, where q is the number of ansatz parameters in the reverse light cone of each $\mathbf{P_i}$. Then, the distribution of equation 2.9 and its first two derivatives are equal to those of a WHRF

$$F_{WHRF}(\boldsymbol{\theta}) = m^{-1} \sum_{i,j=1}^{2^l} w_i J_{i,j} w_j \tag{2.11}$$

with m degrees of freedom, up to an error in distribution on the order of $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t} + \epsilon + \exp(-l)\right)\right)$. Here, $\mathbf{w}$ are points on the hypertorus $\left(S^1\right)^{\times l}$ parameterized by $\tilde{\boldsymbol{\theta}}$, where $\tilde{\theta}_i$ is the sum of all θ_j on qubit i.

We interpret this result as the degrees of freedom m of the model being given by roughly the sum of the local Hilbert space dimensions of the reverse light cones of terms in the Pauli decomposition of $\mathbf{H}$. We interpret this as the local underparameterization of the model, to be contrasted with the global underparameterization interpretation when m is exponentially large in n. Using known properties of the loss landscapes of WHRFs (see Methods), we are then able to prove the following result on the loss landscapes of local VQAs:

Corollary 2.2.2 (Shallow, local VQAs have poor loss landscapes, informal)**.** *Let $\widehat{\mathcal{R}}_{VQE}$ be a local VQA loss function of the form of equation 2.9. Assume all coefficients α_i of the Pauli decomposition of $\mathbf{H}$ are $\Theta(1)$, and*

$$l \log(n) + q \log(q) = \mathrm{o}\left(2^l A\right). \tag{2.12}$$

Then $\widehat{\mathcal{R}}_{VQE}$ has a fraction superpolynomially small in n of local minima within any constant additive error of the ground state energy.

Optimizing loss landscapes where only a superpolynomially small (in n) fraction of the local minima are near the global minimum in energy is expected to be difficult. Indeed, algorithms such as gradient descent would then expect to have to be restarted a superpolynomial number of times before finding a good approximation to the global minimum; we also give heuristic reasons why this should continue to be true for other local optimizers in Supplementary Note 7. Our results stand in stark contrast with the loss landscapes typically found in classical machine learning, where almost all local minima closely approximate the global minimum in function value [92, 82].

In the shallow ansatz regime—where $q, l = \mathrm{O}(\mathrm{polylog}(n))$—and assuming an extensive problem Hamiltonian such that $A = \Omega(n)$, the condition given by equation 2.12 is always satisfied. Interestingly, this is a regime where barren plateaus are known not to occur [256], demonstrating that poor local minima can give rise to poor optimization performance even when the loss function features large gradients. We now specialize to common variational quantum learning scenarios, and consider the implications of Corollary 2.2.2.

First, let us consider d-dimensional checkerboard ansatzes of constant depth. Fix p, t to be sufficiently large constants. We assume that the initial state forms an $\mathrm{O}\left(\frac{1}{\mathrm{poly}(t)}\right)$-approximate t design on l qubits around each Pauli observable of weight k; this can be done via a depth p, d-dimensional circuit of 2-local Haar random unitaries when $l = \mathrm{O}\left(\frac{(p+k)^d}{\mathrm{poly}(t)}\right) \geq$

k for some fixed polynomial in t [161, 158]. After this state preparation circuit, a traditional depth $\Theta\left(l^{\frac{1}{d}}\right)$ (i.e. independent of n), d-dimensional, n qubit checkerboard circuit is applied, with observable reverse light cones of size at greatest l. By Corollary 2.2.2, these variational ansatzes are untrainable due to poor local minima, yet by the results of [256] do not suffer from barren plateaus.

One interesting consideration is extending this result to traditional checkerboard ansatzes, without the special state preparation procedure we have considered. There, the $l = \mathrm{O}\left(\frac{(p+k)^d}{\mathrm{poly}(t)}\right)$ qubit local state is mixed, and our results therefore do not directly apply. However, we expect no reason for the mixedness of the initial state to improve training performance in any way. We validate this intuition numerically in Section 2.2.4.

We also consider a class of models similar to quantum convolutional neural networks (QCNNs) [97] previously shown not to suffer from barren plateaus [275]. Though these models are in full generality trained on arbitrary loss functions, for learning various physical models the loss may take the form of equation 2.4. QCNNs are defined by their measurement of a subset of qubits at periodic intervals, via so-called pooling layers; for sufficiently deep (i.e. large constant depth) convolutional layers, then, at some point in the model, the number of remaining qubits will be sufficiently small such that the remaining convolutional layers are approximately scrambling. If one then assumes that the initial states are adversarially chosen such that they remain pure by this layer, this scenario reduces to the shallow checkerboard ansatz scenario, and once again we expect poor local minima by Corollary 2.2.2. Even if the initial states are not adversarially chosen and the input to the scrambling convolutional layers is mixed, we expect by similar intuition the model to remain untrainable; we will see this numerically, where we also observe that this poor training occurs when training on loss functions beyond equation 2.4.

2.2.4 Numerical Results

To numerically validate our theoretical findings, we perform numerical simulations showing that learning in various settings cannot be guaranteed unless exponentially many parameters are included in an ansatz. We only consider problems and ansatzes where the existence of a zero loss global minima is guaranteed to study whether or not optimizers can actually find the global minimum or a similarly good critical point. We parameterize all trainable 2-qubit gates in the Lie algebra of the 4-dimensional unitary group, and implement the resulting

unitary matrix via the exponential map which is surjective and capable of expressing any local 4×4 unitary gate. In all cases, we perform simulations using calculations with computer precision and analytic forms of the gradient (see Supplementary Note 6 for more details). In practice, actual quantum implementations will be hampered by various sources of inefficiency such as the lack of an analogous method of backpropagation for calculating gradients, sampling noise, or even gate errors. Thus, our numerical analysis can be interpreted as a "best case" setting for quantum computation where we disregard such inefficiencies and focus solely on learnability. In Supplementary Note 5, we further study variations of the teacher-student learning and random variational quantum eigensolver (VQE) [273] settings discussed here. We also consider the training performance of VQE in finding the ground state of a Heisenberg XYZ Hamiltonian [164]. Our supplemental results reinforce our findings here.

One may conjecture that it is plausible to learn the class of functions generated by relatively shallow depth variational teacher circuits by parameterizing a shallow-depth student circuit of the same form and training its parameters. In this so-called teacher-student setup, we are guaranteed the existence of a perfect global minimum since recovering the parameters of the teacher circuit achieves zero loss. In other words, the global minimum is guaranteed to be achievable in the setting we consider here. Still, we showed earlier that such circuits typically have many poor local minima, and are always hard to learn in the statistical query setting. Here, we provide numerical evidence of these findings for the QCNN ansatz. Additional confirmation of these findings with a checkerboard ansatz is included in Supplementary Note 5.

The quantum convolutional neural network (QCNN) presents an interesting test bed for our analysis since it has been shown in prior work to avoid barren plateaus [275]. Nevertheless, the QCNN, like other models, is riddled with poor local minima in generic learning tasks. For our analysis, we attempt to learn randomly generated quantum convolutional neural networks (QCNNs) with a parameterized QCNN of the same form. In the QCNN, both student and teacher circuits have parameterized 2-qubit gates at each layer followed by 2-qubit pooling layers (see Supplementary Note 6 for more details). Each 2-qubit gate is fully parameterized in the Lie algebra of the unitary group. Networks are trained to predict the probability of the measurement of the last qubit in the teacher circuit. In other words, the student network is trained on a classification problem defined by teacher network where, by construction, perfect classification accuracy is known to be achievable. We benchmark

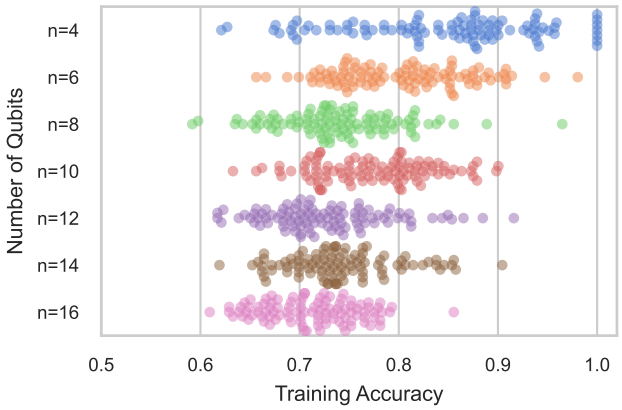


Figure 2-2: **Teacher-student evaluation for the n qubit QCNN.** The student circuit is unable to learn the teacher circuit as the number of qubits grows, converging to a local minimum of the loss landscape. The existence of a global optimum is guaranteed as the teacher circuit is drawn from a random initialization of the same QCNN structure of the student circuit. Here, for a ranging number of qubits, 100 student circuits are trained to learn randomized teacher circuits of the same form and the resulting swarm plots of the final training accuracy are shown.

performance with the classification accuracy, where a prediction is considered correct when it predicts the most likely measurement of the last qubit correctly. Networks are trained via the Adam optimizer [212] to learn outputs of 512 randomly chosen computational basis states. QCNNs with 4, 8, 12, and 16 qubits have 32, 48, 64, and 64 trainable parameters, respectively.

Fig. 2-2 plots the final training accuracy achieved over 100 random simulations for varying ranges of circuit sizes. For circuits with 4 qubits, the training is sometimes successful, often achieving an accuracy above 85 percent on the training dataset. However, as the number of qubits grows, even past 8 qubits, the optimizer is unable to recover parameters which match the outputs of the teacher circuit. The results here show that the QCNN circuit—which has $\mathrm{O}(\log n)$ depth—still scrambles outputs to hinder learnability.

We now consider VQE. To analyze the performance of variational optimizers, we consider problems and ansatzes which are capable of recovering the global minimum. We aim to find the ground states of local Hamiltonians $\boldsymbol{H}_t$ over n qubits that take the form of single qubit Pauli $\boldsymbol{Z}$ Hamiltonians conjugated by L^* layers of two alternating unitary operators $\boldsymbol{U}_1$ and

$\boldsymbol{U}_2$ which are product unitaries on neighboring 2-local qubits:

$$\boldsymbol{H}_t = \left(\boldsymbol{U}_2^\dagger \boldsymbol{U}_1^\dagger\right)^{L^*} \left[\sum_{i=1}^{n} \boldsymbol{Z}_i\right] (\boldsymbol{U}_1 \boldsymbol{U}_2)^{L^*} + n\boldsymbol{I}. \tag{2.13}$$

The added identity matrix normalizes the Hamiltonian to have ground state with energy 0. Since the ground state of $\sum_{i=1}^{n} \boldsymbol{Z}_i$ is the state $|1\rangle^{\otimes n}$, we are guaranteed the existence of a global minima when using a checkerboard ansatz of at least depth L^*, since this ansatz can "undo" the conjugation by unitary operators. In the remainder of this Section, we consider equation 2.13 with $L^* = 4$.

We measure the performance of optimization with two metrics. The first is the loss function itself, which is the average energy $\langle\psi| \boldsymbol{H}_t |\psi\rangle$ of the VQE ansatz state $|\psi\rangle$ for the given Hamiltonian $\boldsymbol{H}_t$. The second is the trace distance to the ground state $|\phi_g\rangle$ of $\boldsymbol{H}_t$, equal to $\| |\phi_g\rangle\langle\phi_g| - |\psi\rangle\langle\psi| \|_1/2$. Both of these metrics converge to zero at the global minimum.

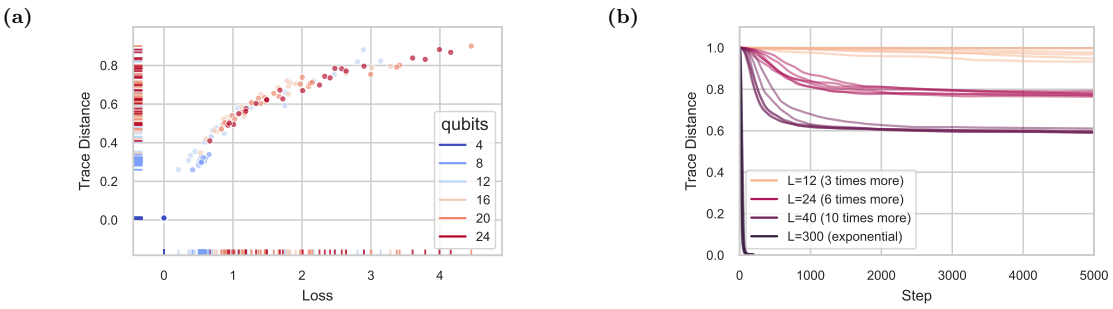


Figure 2-3: **Empirical analysis of VQE.** (a) Scatter plot of the final loss and trace distance of the VQE state after 30000 steps of gradient descent optimization shows that the algorithm converges to poorer local minima as the number of qubits grows. 24 simulations are performed for each value of n. The algorithm always succeeds at obtaining the ground state with 4 qubits, but progressively struggles more with added qubits. (b) The number of layers needed to guarantee convergence to the ground state empirically grows exponentially with the number of qubits. Here, we consider 4-layer Hamiltonians of the form of equation 2.13 on 14 qubits where the number of layers L in the ansatz is varied. When the ansatz has 300 layers—enough that the number of ansatz parameters is larger than the explored Hilbert space dimension—the model successfully converges to the ground state, rather than remaining stuck in a poor local minimum.

We first aim to learn the ground state using a checkerboard ansatz by performing vanilla gradient descent on $L = L^* = 4$ parameterized layers, equal in depth to the Hamiltonian conjugation circuit and thus capable of recovering the ground state. In Fig. 2-3(a), we plot the final values of the loss and trace distance for 24 randomly initialized VQE problems for

a number of qubits ranging from 4 to 24. Similar results are observed when using more advanced optimizers such as Adam (see Supplementary Note 6) [212]. Consistent with our theoretical findings, convergence clusters around local minima far from the ground state, particularly as the number of qubits grows.

Our theoretical results also imply the difficulty of training beyond a finite fraction of the ground state energy in a VQE setting. Fig. 2-3(b) illustrates this phenomenon when performing optimization on a 14 qubit ansatz. As more parameters are added to the ansatz via increasing its depth L, the VQE algorithm performs better, but it is not until the number of parameters is exponential in the problem size that convergence to a global minimum (or even within a small additive error of the global minimum) is guaranteed. This is true even though the ansatz is capable of expressing the ground state at $L = 4$. Simulations here are performed as before on random $L^* = 4$ Hamiltonians of the form of equation 2.13.

2.3 Discussion

Though variational quantum algorithms—and quantum machine learning models in general—have been cited as perhaps the most promising use case for quantum devices in the near future [278], theoretical guarantees of their training performance have been sparse. Here, we have excluded a wide class of variational algorithms by showing that in many settings, they are in fact not trainable. We showed this in two different frameworks: first, in Section 2.2.2, we studied various classes of quantum models in the statistical query framework. We showed that in the presence of noise, exponentially many queries in the problem size are needed for these models to learn. As a complementary approach, we also examined the typical loss landscapes of variational quantum algorithms in the noiseless setting in Section 2.2.3, and showed that even at constant depth these models can have a number of poor local minima superpolynomially large in the problem size. We also numerically confirmed these results for a variety of problems in Section 2.2.4. These results go beyond the typical studies on the presence of barren plateaus, as many of the models we study here have gradients vanishing only polynomially quickly in the problem size. Our work demonstrates that showing that barren plateaus are not present in a model does not necessarily vindicate it as trainable.

These results, though they exclude a wide variety of variational quantum algorithms, still leave room for hope in the usefulness of these algorithms. Particularly, our analysis in the

noiseless setting of landscapes of variational quantum algorithms focuses on very general, Hamiltonian agnostic ansatzes; in various instances, more focused ansatzes may be trainable. For instance, as previously shown in [129], for certain classes of problems the quantum approximate optimization algorithm (QAOA) [127] is provably able to outperform the best unconditionally proven classical algorithms, even when taking into account the training of the model. This is due to parameter concentration, where the global optimum for small problem instances is close to the global optimum for large problem instances [56]. These results demonstrate the power of good model initialization in variational quantum algorithms: even if the total variational landscape is swamped with poor local minima, good initialization may ensure that the optimizer begins in the region of attraction of the global minimum. Though this is perhaps most relevant for the variational quantum eigensolver (VQE) [273] and QAOA [127], where there exists physical intuition for potentially performant parameter initializations, in more traditional machine learning settings this may manifest as good performance on certain inputs to the model.

Variationally studying models with many symmetries may also avoid our poor performance guarantees. Intuitively, our results here are the consequence of underparameterization. Namely, unless the ansatz is parameterized such that the number of parameters grows with the (local) Hilbert space dimension, the model is not trainable. Typically, this Hilbert space dimension is exponentially larger than the number of parameters the ansatz uses to explore it. However, if the model is heavily constrained by symmetries, this dimension might be much smaller. Such models were studied numerically in [217, 335], where it was shown that certain variational quantum algorithms optimize efficiently. Though often these models can be solved classically when the symmetries are known, these symmetries may not be known *a priori*. Indeed, one may be able to test for the presence of symmetries in a given model by studying whether associated variational quantum algorithms are trainable. Similar to these general symmetry considerations, known structure in the problem may also allow one to build up hierarchical ansatzes that are able to be trained sequentially. We leave further investigation in these directions to future work.

Finally, though many variational models fit the framework of equation 2.4, there exist other settings of variational quantum algorithms. One class of such models includes quantum Boltzmann machines, which attempt to model given quantum states via the training of quantum Gibbs states [11]. When the full quantum Gibbs state is observed, it is known that

these models are efficiently trainable [21], and numerically it is known that these models are trainable even when the full state is not observed [11, 208]. Furthermore, though in full generality preparing quantum Gibbs states is difficult, state preparation has been shown to be efficient in certain regimes relevant to machine learning [208, 16, 357], potentially giving an end-to-end trainable quantum machine learning model. We leave further analytical investigation on the training landscapes of quantum Boltzmann machines to future work.

Our results contribute to the already vast library of literature on the trainability of variational quantum models in further culling the landscape of potentially trainable quantum models. We hope these results have the effect of focusing research efforts toward classes of models that have the potential for trainability, and whittle down the search for practical use cases of variational quantum algorithms.

2.4 Methods

2.4.1 The Statistical Query Learning Framework

We give a brief overview of the classical SQ model here, and provide a more detailed review in Supplementary Note 2. Given an input and output space $\mathcal{X}$ and $\mathcal{Y}$, let $\mathcal{D}$ be a joint distribution on $\mathcal{X} \times \mathcal{Y}$. In the classical SQ model, one queries the SQ model by inputting a function f and receiving an estimate of $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(x,y)]$ within a given tolerance τ. As an example, one can query a loss function ℓ for a model $m_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ by querying the function $\ell(m_{\boldsymbol{\theta}}(x), y)$. A special class of statistical queries are inner product queries where query functions g are defined only on $\mathcal{X}$ and the correlational statistical query returns an estimate of $\mathbb{E}_{(x,y)\sim\mathcal{D}}[g(x)\cdot y]$ within a specified tolerance τ.

In detail, the SQ models we consider take the forms below:

Definition 2.4.1 (Quantum correlational statistical query (qCSQ))**.** Assume there is a target observable $\boldsymbol{M}$ that we would like to learn on some distribution over states $\mathcal{D}$. Applying the correlational SQ model to the quantum setting, we define the query $\mathrm{qCSQ}(\boldsymbol{O}, \tau)$ which takes in a bounded observable $\boldsymbol{O}$ with $\|\boldsymbol{O}\| \leq 1$ and a tolerance τ and returns a value in the range:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho) - \tau\right] \leq \mathrm{qCSQ}(\boldsymbol{O},\tau) \leq \mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho) + \tau\right]. \tag{2.14}$$

Definition 2.4.2 (Quantum unitary statistical query (qUSQ))**.** In the unitary compilation setting, one aims to learn a target unitary transformation $\boldsymbol{U}_*$ over a distribution $\mathcal{D}$ of input/output pairs of that unitary transformation. Here, the oracle $\mathrm{qUSQ}(\boldsymbol{V}, \tau)$ takes in a unitary matrix $\boldsymbol{V}$ and a tolerance τ and returns a value in the range:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V}\rho)] - \tau\right] \leq \mathrm{qUSQ}(\boldsymbol{V}, \tau) \leq \mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V}\rho)] + \tau\right]. \tag{2.15}$$

Importantly, if $\mathcal{D}$ is a 1-design over n qubit states, then the above can be simplified using the formula $\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V}\rho)]\right] = 2^{-n}\Re\left[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V})\right]$ (see proof in Supplementary Note 3). Queries to qUSQ are related to performing a Hadamard test [7], also a common subroutine in variational algorithms [330].

The queries above take the forms of inner products, with $\langle \boldsymbol{M}_1, \boldsymbol{M}_2\rangle_\mathcal{D} = \mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{M}_1\rho)\,\mathrm{Tr}(\boldsymbol{M}_2\rho)\right]$ and $\langle \boldsymbol{U}_1, \boldsymbol{U}_2\rangle_\mathcal{D} = \mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re\left[\mathrm{Tr}(\boldsymbol{U}_1^\dagger \boldsymbol{U}_2\rho)\right]\right]$. The inner products also induce corresponding L_2 norms: $\|\boldsymbol{M}\|_\mathcal{D} = \sqrt{\langle \boldsymbol{M}, \boldsymbol{M}\rangle_\mathcal{D}}$. As the magnitude of this norm can change with the dimension, we introduce the quantity C_{max} to denote the maximum value a query can take for any target observable in the qCSQ model, i.e. $C_{max} = \max_{\boldsymbol{M}:\|\boldsymbol{M}\|\leq 1} \|\boldsymbol{M}\|_\mathcal{D}^2$. For fair comparison, we quantify noise tolerances and hardness bounds with respect to C_{max}. Note that for the qUSQ model $C_{max} = 1$, but in the qCSQ model, C_{max} can decay with the number of qubits under for example the Haar distribution of inputs.

A statistical query algorithm learns a function class if it can output a unitary or observable that is close to any target in that class.

Definition 2.4.3 (qCSQ / qUSQ learning of hypothesis class)**.** A given algorithm using only statistical queries to qCSQ (qUSQ) successfully learns a hypothesis class $\mathcal{H}$ consisting of observables $\boldsymbol{M}, \|\boldsymbol{M}\| \leq 1$ (unitaries $\boldsymbol{U}$) up to ϵ error if it is able to output an observable $\boldsymbol{O}$ (unitary $\boldsymbol{V}$) which is ϵ-close to the unknown target observable $\boldsymbol{M} \in \mathcal{H}$ ($\boldsymbol{U} \in \mathcal{H}$) in the L_2 norm, i.e., $\|\boldsymbol{M} - \boldsymbol{O}\|_\mathcal{D} \leq \epsilon$ ($\|\boldsymbol{U} - \boldsymbol{V}\|_\mathcal{D} \leq \epsilon$).

The statistical query dimension quantifies the complexity of a hypothesis class $\mathcal{H}$ and is related to the number of queries needed to learn functions drawn from a class, as summarized in Theorem A.3.3.

Definition 2.4.4 (Statistical query dimension [53, 283])**.** For a distribution $\mathcal{D}$ and concept class $\mathcal{H}$ where $\|\boldsymbol{M}\|_\mathcal{D}^2 \leq C_{max}$ for all $\boldsymbol{M} \in \mathcal{H}$, the statistical query dimension ($\mathrm{SQ\text{-}DIM}_\mathcal{D}(\mathcal{H})$)

is the largest positive integer d such that there exists d observables $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_d \in \mathcal{H}$ such that for all $i \neq j : |\langle \boldsymbol{M}_i, \boldsymbol{M}_j \rangle_{\mathcal{D}}| \leq C_{max}/d$.

Theorem 2.4.5 (Query complexity of learning [312, 53])**.** *Given a distribution $\mathcal{D}$ on inputs and a hypothesis class $\mathcal{H}$ where $\|\boldsymbol{M}\|_{\mathcal{D}}^2 \leq C_{max}$ for all $\boldsymbol{M} \in \mathcal{H}$, let $d = \text{SQ-DIM}_{\mathcal{D}}(\mathcal{H})$ be the statistical query dimension of $\mathcal{H}$. Any* qCSQ *or* qUSQ *learner making queries with tolerance $C_{max}\tau$ must make at least $(d\tau^2 - 1)/2$ queries to learn $\mathcal{H}$ up to error $C_{max}\tau$.*

Since our setting differs slightly from the standard classical setting [312, 53], we include a proof of the above in Supplementary Note 3. For example, if the hypothesis class is rich enough to be able to express any n-qubit Pauli observable, then the statistical query dimension of that class is at least 4^n over the Haar distribution of inputs since Pauli observables are all orthogonal. This forms the basis for our resulting proofs of hardness, summarized in Table 2.1 and proved in Supplementary Note 3.

Analogous to work in classical machine learning [139], one can perform noisy gradient descent as a series of statistical queries. As an example, consider the task of learning a target Hamiltonian $\boldsymbol{M}$ by constructing a variational Hamiltonian $\boldsymbol{H}(\boldsymbol{\theta}) = \boldsymbol{U}(\boldsymbol{\theta})^\dagger \boldsymbol{H} \boldsymbol{U}(\boldsymbol{\theta})$ with parameterized Pauli rotations and minimizing the mean squared error between expectations of $\boldsymbol{M}$ versus $\boldsymbol{H}(\boldsymbol{\theta})$ over a distribution of states $\mathcal{D}$. Our loss function is

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\rho \sim \mathcal{D}} \left[\left(\text{Tr}[\boldsymbol{M}\rho] - \text{Tr}[\boldsymbol{H}(\boldsymbol{\theta})\rho] \right)^2 \right]. \tag{2.16}$$

The parameter shift rule [293] provides a means to calculate the partial derivative of a function $f(\mu)$ with respect to a parameter μ applied as a parameterized quantum gate $e^{-i\mu \boldsymbol{G}}$ by calculating the function itself at two shifted coordinates. For example, for parameterized Pauli gates ($\boldsymbol{G} \in \frac{1}{2}\{\boldsymbol{Z}, \boldsymbol{X}, \boldsymbol{Y}\}$), this takes the form:

$$\frac{\partial}{\partial \mu} f(\mu) = \frac{1}{2} \left[f\left(\mu + \frac{\pi}{2}\right) - f\left(\mu - \frac{\pi}{2}\right) \right]. \tag{2.17}$$

By applying the parameter shift rule [293], we can evaluate the gradient of the loss with respect to parameter entry θ_i as

$$\frac{\partial}{\partial \theta_i} \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\rho \sim \mathcal{D}} \left[\left(\text{Tr}[\boldsymbol{H}(\boldsymbol{\theta})\rho] - \text{Tr}[\boldsymbol{M}\rho] \right) \left(\text{Tr}[\boldsymbol{H}(\boldsymbol{\theta}^+)\rho] - \text{Tr}[\boldsymbol{H}(\boldsymbol{\theta}^-)\rho] \right) \right], \tag{2.18}$$

where θ^+ and θ^- are the values of the parameters shifted at the i-th entry according to the parameter shift rule for the gradient. The quantity $\mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}[\boldsymbol{H}(\theta)\rho]\big(\mathrm{Tr}[\boldsymbol{H}(\theta^+)\rho]-\mathrm{Tr}[\boldsymbol{H}(\theta^-)\rho]\big)\right]$ can be directly evaluated without statistical queries, and the quantity $\mathbb{E}_{\rho\sim\mathcal{D}}[\mathrm{Tr}[\boldsymbol{M}\rho]\big(\mathrm{Tr}[\boldsymbol{H}(\theta^+)\rho]-\mathrm{Tr}[\boldsymbol{H}(\theta^-)\rho]\big)]$ can be evaluated using 2 statistical queries to qCSQ where the tolerance τ accounts for the noise in the estimate.

As a second example, this time in the unitary compiling setting of qUSQ, we can evaluate the commonly used procedure of measuring the inner product or average fidelity of n-qubit states between a target unitary $\boldsymbol{U}_*$ and a variationally chosen unitary $\boldsymbol{V}(\theta)$ using statistical queries analogous to a swap test on actual quantum hardware [250, 60, 197, 38]. With slight abuse of notation, let $|\phi\rangle\sim\mathcal{D}$ denote a distribution over pure states which forms a 2-design. Then via averaging over 2-designs (see Supplementary Note 3 for details), the average fidelity equals

$$\mathbb{E}_{|\phi\rangle\sim\mathcal{D}}\left[F\Big(\boldsymbol{U}_*|\phi\rangle,\boldsymbol{V}(\theta)|\phi\rangle\Big)\right]=\mathbb{E}_{|\phi\rangle\sim\mathcal{D}}\left[|\langle\phi|\boldsymbol{V}(\theta)^\dagger\boldsymbol{U}_*|\phi\rangle|^2\right]=\frac{2^{-n}\left|\mathrm{Tr}(\boldsymbol{V}(\theta)^\dagger\boldsymbol{U}_*)\right|^2+1}{2^n+1}. \tag{2.19}$$

Note, that the key quantity $\left|\mathrm{Tr}(\boldsymbol{V}(\theta)^\dagger\boldsymbol{U}_*)\right|^2=\Re[\mathrm{Tr}(\boldsymbol{V}(\theta)^\dagger\boldsymbol{U}_*)]^2+\Re[i\,\mathrm{Tr}(\boldsymbol{V}(\theta)^\dagger\boldsymbol{U}_*)]^2$ can be evaluated up to a desired tolerance using statistical queries $\mathrm{qUSQ}(\boldsymbol{V}(\theta),\tau)$ and $\mathrm{qUSQ}(i\boldsymbol{V}(\theta),\tau)$.

One important caveat must be noted that in the SQ setting, learning must succeed for all values of the query within the given tolerance τ. Noise in quantum settings, which can arise from sampling a finite data set, gate error, state preparation error, measurement sampling noise, or other means does not exactly coincide with the assumed tolerance of an SQ model. Nevertheless, though noise during optimization may appear unnatural in classical settings, such noise in quantum settings is rather endemic and the SQ model allows one to rigorously analyze the complexity of learning in the presence of noise.

2.4.2 The Loss Landscapes of Wishart Hypertoroidal Random Fields

The loss landscapes of Wishart hypertoroidal random fields (WHRFs; see Supplementary Note 4 for a brief review) are known [20] to exhibit a computational phase transition governed by the order parameter

$$\gamma=\frac{l}{2m}, \tag{2.20}$$

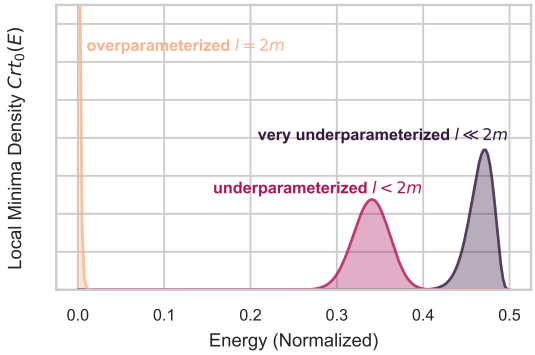


Figure 2-4: **Characteristic distribution of local minima.** Plot of the asymptotic distribution of local minima of WHRFs with m degrees of freedom on the l-torus in: the extremely underparameterized regime, where $l \ll 2m$; the moderately underparameterized regime, where l is a finite fraction of $2m$; and at the critical overparameterization regime, where $l = 2m$. Here, the energy is scaled and shifted as per equation 2.9 so that global minima have zero energy. In the underparameterized regime, only a fraction $\sim \exp(-m)$ of the critical points are within any constant additive error of the global minimum. In the overparameterized regime, local minima are exponentially concentrated at the global minimum.

called the overparameterization ratio. Here, l is the number of parameters of the WHRF, and m its degrees of freedom (see Supplementary Note 4). When $\gamma \ll 1$ (the underparameterized regime), WHRFs exhibit poor local minima and thus are essentially untrainable; when $\gamma \geq 1$ (the overparameterized regime), however, essentially all local minima of a WHRF are close to the global minimum in function value. More specifically, when $\gamma = \mathrm{o}\left(\frac{1}{\log(n)}\right)$, a superpolynomially small (in n) fraction of the local minima are within any constant additive energy error to the global minimum. When restoring units to the variational risk of equation 2.9, this is an error extensive in the problem size. The asymptotic expression of the distribution of local minima is also known, which is given by (up to a normalization factor):

$$\mathrm{Crt}_0(E) \sim \mathrm{e}^{-mE} E^{m-l/2} (1-2E)^l \tag{2.21}$$

for the density of local minima at any given energy $0 \leq E \leq \frac{1}{2}$, in units of the mean eigenvalue of $\mathbf{H}$ (shifted such that the global minimum is at $E = 0$). Representative plots of this distribution in various parameterization regimes are shown in Fig. 2-4.

This distribution of local minima is calculated from the joint distribution of the WHRF function value, its gradient, and its Hessian. Thus, by demonstrating the convergence of this

joint distribution in the variational loss functions we consider to the analogous distribution in WHRFs at a sufficient rate, we are able to show the same phase transition occurs in variational loss functions. Our full proof is given in Supplementary Note 4.

Data Availability

The processed data generated and analysed for this study are available at https://github.com/bkiani/Beyond-Barren-Plateaus and [200].

Code Availability

The code used for the current study is available at [200].

Chapter 3

Quantum Wasserstein GAN

The previous chapter showed that care must be taken in the design of quantum machine learning algorithms for them to work. Here, we consider the question of which distance metrics to use when attempting to learn data that takes the form of quantum states. As we will see, the inner product and distances derived from the inner product are not good choices in learning settings as such a distance metric on average exponentially decays with the number of qubits. This motivates us to consider a different distance metric based on optimal transport theory which does not feature such exponential decay.

3.1 Introduction

A fundamental task in quantum machine learning is designing efficient algorithms for learning quantum states [42, 100, 314, 134, 1, 284, 237, 71, 79, 38], transformations [206, 250, 51, 280, 236, 70, 38, 299], and classical data stored as or generated by quantum states [41, 231, 99]. In the general setup, one is given a target quantum object, say a target quantum state, and aims to generate or approximate that target object by efficiently learning parameters in a quantum circuit. For example, quantum generative adversarial networks (qGAN) are parameterized sets of quantum circuits and quantum operators designed to learn target states or transformations via optimization over the parameters of a quantum generator and discriminator [237, 79].

A crucial component of a quantum machine learning algorithm is an objective function (often a distance metric) which determines how close a generated object is to its target. This choice of metric is important not only as a measure of performance but also as a means for

optimization. For example, certain metrics lead to efficient algorithms for calculating gradients with respect to that metric, allowing algorithms to perform optimization via gradient based optimizers (*e.g.,* gradient descent). Naturally, for learning pure states, a common choice for the distance metric is a function of the inner product between quantum states. Similarly, for learning density matrices, researchers commonly choose distance metrics which simplify to a function of the inner product when measuring the distance between pure states.

Previous approaches to learning quantum data have typically suffered from the presence of vanishing gradients [246, 329, 78] and poor local minima [272, 253, 75] in the loss landscape induced by the choice of distance metric. Intuitively, these "barren plateaus" and traps arise due to the fact that random quantum states have inner product that diminishes exponentially with the number of qubits. Our approach helps surmount these challenges by formulating an algorithm which provides an efficient means for learning pure and mixed states using the recently proposed quantum earth mover's (EM) distance, also known as the quantum Wasserstein distance of order 1 [108]. As we will demonstrate, the quantum EM distance is a natural distance metric for optimization over local operations and avoids common pitfalls faced by other distance metrics which reduce to functions of the inner product. This is in agreement with results in classical machine learning, where algorithms employing the earth mover's distance are often more stable and avoid issues with vanishing or exploding gradients [23, 84, 289, 322, 155] (see section B.1 for a more complete discussion of the literature and section B.2 for a presentation of the classical EM distance). Intuitively, the quantum EM distance can be interpreted as a continuous version of a quantum "Hamming distance", which allows local gates to optimize over the few qubits on which they act instead of over some global distance metric which often decays exponentially in the number of qubits.

In this work, we study the quantum EM distance from an applied setting and make the following contributions. First, we overview the construction of the quantum EM distance and analyze the properties of different quantum distance metrics and the loss landscapes they produce in quantum machine learning settings. Here, we show that the quantum EM distance has unique advantages over other common distance metrics. Then, to operationalize the quantum EM distance, we devise a new heuristic method to approximate the quantum EM distance efficiently given copies of quantum states. In learning settings, this leads to our development of a quantum Wasserstein generative adversarial network (qWGAN) which is a quantum analog to the classical Wasserstein generative adversarial network [23] (see also

[79]). Importantly, like its classical analog, our qWGAN employs an earth mover's distance in its cost function. Numerical results show that our qWGAN is efficient at learning quantum data with shallow circuits in various settings. Finally, we discuss near term applications of our qWGAN for both classical and quantum problems.

3.2 Quantum distance metrics and quantum EM distance

To approximate or reconstruct a target probability distribution with a machine learning algorithm, the choice of distance metric, measuring how well the approximating distribution matches the target distribution, is crucial to the performance of the algorithm. Classically, generative adversarial networks (GAN) provide a neural network approach for learning a target probability distribution and generating new samples from the approximate distribution [147, 23]. The choice of loss metric for a GAN is a distance or divergence metric which is minimized when the target and generated distributions coincide.

In the quantum setting, distance metrics between states or density matrices are employed in the implementation of quantum generative adversarial networks (qGAN) [173, 79, 237, 75]. As in the classical setting, the choice of distance metric is crucial to the runtime and performance of the quantum machine learning algorithm. Here, we consider common distance metrics and show that the quantum earth mover's (EM) distance recently defined in [108] possesses desirable properties that are not found in the other metrics.

For a brief overview of the notation used in quantum mechanics, we refer the reader to section B.3. Let $\rho, \sigma \in \mathbb{C}^{N\times N}$ be the density matrices corresponding to two quantum states, *e.g.,* ρ can be the quantum state generated by a GAN and σ is the target state. Until now, common distance metrics employed to train quantum GANs have been unitarily invariant, *i.e.*, invariant with respect to the conjugation of both quantum states with the same unitary matrix and reducing to a function of the inner product for pure states (*i.e.,* orthogonal projectors with rank one). Commonly used distance metrics in prior works include:

- **Trace Distance:** The simplest and most common choice (*e.g.,* see [173, 42]) is the trace distance:

$$D_1(\rho, \sigma) = \frac{1}{2} \|\rho - \sigma\|_1 \ , \tag{3.1}$$

  where $\|\cdot\|_1$ denotes the trace norm, *i.e.*, the sum of the singular values.

- **Quantum Fidelity:** Another common choice (*e.g.,* see [38]) is the maximum absolute value squared of the inner product between purifications of ρ and σ:

$$F(\rho, \sigma) = \|\sqrt{\rho}\,\sqrt{\sigma}\|_1^2 \,. \tag{3.2}$$

  $F(\rho, \sigma)$ is often modified to $\arccos\sqrt{F(\rho, \sigma)}$ to construct a proper distance metric.

- **Quantum Wasserstein Semimetric:** Introduced in [79] as a quantum generalization of the Wasserstein distance, this distance, denoted $qW(\rho, \sigma)$, is calculated by forming a coupling between quantum states ρ and σ in $\mathbb{C}^{N\times N}$. The coupling is a quantum state in $\left(\mathbb{C}^{N\times N}\right)^{\otimes 2}$ whose marginal states are equal to ρ and σ, respectively. The quantum Wasserstein semimetric is the minimum of the expectation value of the projector onto the symmetric subspace of $\left(\mathbb{C}^{N}\right)^{\otimes 2}$. qW does not satisfy the triangle inequality, hence the name semimetric. Importantly, qW is unitarily invariant, and for pure states, it reduces to a function of their inner product: for any $|u\rangle\,, |v\rangle$ unit vectors in $\mathbb{C}^N$, $qW(|u\rangle\langle u|\,, |v\rangle\langle v|) = (1 - |\langle v|u\rangle|^2)/2$. Further details can be found in section B.4.

The quantum EM distance In this paper we consider the case of n qubits, where $N = 2^n$, and employ the quantum generalization of the Wasserstein distance of order 1 to the states of n qubits recently proposed in [108] and also known as the earth mover's (EM) distance. We adopt the latter terminology as it is more prevalent in the machine learning community, hereby denoting the quantum EM distance with D_{EM}. Unlike all the previously employed distances, the quantum EM distance is not unitarily invariant. We will show that, similar to its classical counterpart [23], D_{EM} possesses several properties that are desirable when learning quantum data.

The quantum EM distance of [108] is based on the notion of neighboring states. Two quantum states of n qubits are neighboring if they differ in only one qubit, *i.e.*, if they coincide after one qubit is discarded. The quantum EM distance is the distance that is induced by the maximum norm that assigns distance at most one to any couple of neighboring states. We denote with $\|\cdot\|_{EM}$ the corresponding norm, whose analytical expression can be found below. This definition enforces the continuity of the distance with respect to local operations, *i.e.*, any quantum operation acting on a single qubit can displace a state by at

most one unit with respect to the quantum EM distance. Indeed, for the quantum states of the computational basis the quantum EM distance recovers the classical Hamming distance (equal to the number of elements that differ between two strings), *i.e.*, for any two strings of n bits x and y we have $D_{EM}(|x\rangle\langle x|, |y\rangle\langle y|) = h(x, y)$. More generally, for quantum states diagonal in the computational basis, the quantum EM distance recovers the classical EM distance. The quantum EM distance admits a dual formulation [108], based on the quantum generalization of the Lipschitz constant, which is more suitable for implementation of quantum GANs.

We denote with $\mathcal{O}_n$ the set of n-qubit observables, *i.e.*, the set of the $2^n \times 2^n$ Hermitian matrices. The quantum Lipschitz constant of the observable $H \in \mathcal{O}_n$ is

$$\|H\|_L = 2 \max_{i=1,\ldots,n} \min \left\{ \|H - \tilde{H}_{\bar{i}}\|_\infty : \tilde{H}_{\bar{i}} \in \mathcal{O}_n \text{ acts as identity on the } i\text{-th qubit} \right\} . \quad (3.3)$$

The quantum Lipschitz constant defined above is a generalization of the Lipschitz constant for the functions on strings of n bits, and coincides with the classical Lipschitz constant for the observables that are diagonal in the computational basis [108]. The quantum EM distance between the quantum states ρ and σ is equal to the maximum difference between the expectation values on ρ and σ of a quantum observable with Lipschitz constant at most one:

$$D_{EM}(\rho, \sigma) = \max \{ \mathrm{Tr} [(\rho - \sigma) H] : H \in \mathcal{O}_n, \ \|H\|_L \leq 1 \} . \quad (3.4)$$

When the quantum EM distance plays the role of a cost function in a machine learning algorithm, it can be considered as an energy associated to the parameter configuration. For this reason, we may refer to the observables H in equation 3.4 as Hamiltonians.

Equivalently, the quantum EM distance can be also defined by its primal formulation [108, Definition 6]:

$$D_{EM}(\rho, \sigma) = \frac{1}{2} \min \left\{ \sum_{i=1}^{n} \|X_i\|_1 : X_i \in \mathcal{O}_n, \ \mathrm{Tr}_i X_i = 0 \ \forall \, i = 1, \ldots, n, \ \sum_{i=1}^{n} X_i = \rho - \sigma \right\} . \quad (3.5)$$

To show why D_{EM} possesses desirable properties, we first consider the case where both the target σ and the generated ρ are pure states in a simple toy model. Here, as we will show, undesirable critical points are clearly present and endemic to the loss landscapes for metrics which are a function of the inner product between two pure states. In contrast, the

quantum EM distance D_{EM} avoids these undesirable critical points. Finally, we generalize the findings of this toy model to a larger class of quantum machine learning settings.

3.2.1 A simple toy model

In this section, we consider an intuitive example which shows the advantages of using the quantum EM distance when the learning is performed over local quantum gates. Namely, we show that the commonly used distance metrics which are a function of the inner product between states feature two key issues in learning via local gates. First, the inner product between a generated and target state fails to show improvement when gates are optimized one by one or layer-wise [65]. Second, when parameters are initialized randomly, gradients in this example decay exponentially when the distance metric is a function of the inner product. The quantum EM distance avoids both of these drawbacks and allows for efficient learning in this scenario.

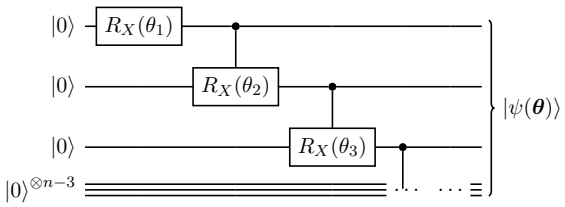
In this toy model, the task at hand is to learn the correct values of parameters in the circuit in Figure 3-1a to generate the GHZ state of n qubits $|GHZ_n\rangle = (|0_n\rangle + |1_n\rangle)/\sqrt{2}$. This circuit consists of a parameterized Pauli X rotation on the first qubit and controlled parameterized Pauli X rotations on later qubits (see section B.3 for description of Pauli operators). When n is a multiple of 4, setting θ_1 equal to $\pi/2$ and all other parameters equal to π will construct the target GHZ state.

Consider the case where one aims to maximize the fidelity F between the generated state $|\psi(\boldsymbol{\theta})\rangle$ and the GHZ state $|GHZ_n\rangle$. Given our circuit, F takes a simple form:

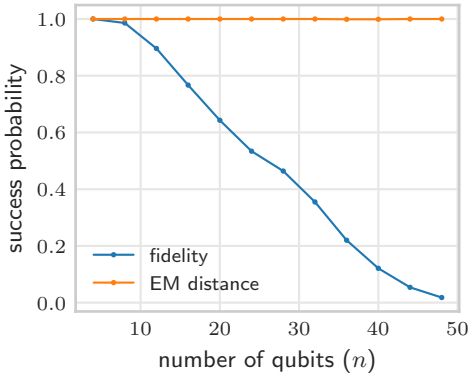
$$F = |\langle GHZ_n|\psi(\boldsymbol{\theta})\rangle|^2 = \left(\frac{\cos(\theta_1)}{\sqrt{2}} + \frac{\prod_{i=1}^{n}\sin(\theta_i)}{\sqrt{2}}\right)^2. \tag{3.6}$$

The first problem associated with learning via F is that the loss landscape has undesirable local minima associated with the state $|0_n\rangle$. Note that fixing any $\theta_i = 0$ will force a learning algorithm (*e.g.,* gradient descent) to optimize the $\cos(\theta_1)$ term, converging to the state $|0_n\rangle$. In other words, if any algorithm aims to optimize the gates in a layer-wise fashion (*e.g.,* optimizing $\theta_1, \theta_2, \ldots$ in order as in [305]), that algorithm will get stuck in the local optimum at $|0_n\rangle$.

Of course, in practice, parameters $\boldsymbol{\theta}$ are typically initialized randomly so it is unlikely that any $\theta_i = 0$; however, even here, we have the second issue that gradients with respect to



(a)



(b)

Figure 3-1: (a) Simple quantum circuit which can generate the GHZ state when parameter values are appropriately chosen. (b) In learning settings, this circuit presents challenges for loss metrics that are a function of the inner product between target and initial states. Simulations show that, with a loss function such as fidelity (blue line), gradient based optimizers eventually fail to find global optimum in virtually all instances when circuit contains many qubits, instead converging to the state $|0_n\rangle$. In contrast, the use of the quantum EM distance (orange line) results in convergence to the global optimum in virtually all instances tested. Here, optimization is performed for up to 10^5 steps using the Adam optimizer (simulations with EM distance generally achieve convergence within about 1000 steps). Experiments are repeated 1000 times for each n to estimate success probability. See section B.5 for full details of experiments.

$\theta_2, \theta_3, \ldots, \theta_n$ all decay exponentially with n since the product of sine functions with random parameter values decays exponentially to zero.

$$\frac{\partial F}{\partial \theta_i} = \begin{cases} -\cos\theta_1 \sin\theta_1 + \cos 2\theta_1 \prod_{k=2}^{n} \sin\theta_k & i = 1 \\ +\sin\theta_1 \cos\theta_1 \left(\prod_{k=2}^{n} \sin\theta_k\right)^2 & \\ \cos\theta_i \prod_{k\neq i} \sin(\theta_k) \left[\cos(\theta_1) + \prod_{k=1}^{n} \sin(\theta_k)\right] & i > 1 \end{cases} . \tag{3.7}$$

Notably, $\frac{\partial F}{\partial \theta_i} = O(\frac{1}{2^n})$ for $i > 1$, but $\frac{\partial F}{\partial \theta_i} = O(1)$ for $i = 1$. For large n, $\frac{\partial F}{\partial \theta_1} \approx -\cos\theta_1 \sin\theta_1$ indicating that gradient optimizers will converge to a poor local minimum outputting the state $|0_n\rangle$ ($\theta_1 = 0 \mod 2\pi$). In fact, since the loss function F (equation equation 3.6) takes a simple form, gradient descent on the parameters can be efficiently performed classically, and results shown in Figure 3-1b show that gradient descent converges to the undesirable local optimum associated with the $|0_n\rangle$ state even as more qubits are added (simulations stopped after 100000 steps of optimization).

The challenges described above are encapsulated by the feature of the inner product that, for example, the states $|0000\rangle$, $|1000\rangle$, and even $|1110\rangle$ are equally distant (orthogonal) to the state $|1111\rangle$ – *i.e.,* local updates induce no change in the inner product distance metric. Using a loss function with the quantum EM distance D_{EM} naturally avoids these challenges. Since the quantum EM distance recovers the Hamming distance between two computational basis states, local operations on one and two qubits can reduce D_{EM} as long as those operations reduce the Hamming distance between the target and generated states. For example, unlike the inner product distance metric, $|1000\rangle$ and $|1110\rangle$ are three and one units away respectively from the state $|1111\rangle$ in the quantum EM distance. In our toy model, local gates, even when applied in isolation, result in changes to single qubits which either reduce or increase the quantum EM distance.

If we set $\theta_1 = \pi/2$, $\theta_2 = \ldots = \theta_k = \pi$ and $\theta_{k+1} = \ldots = \theta_n = 0$ in the quantum circuit of Figure 3-1a, we obtain the quantum state $|\Psi_k\rangle = \left(|0_k\rangle + (-i)^k |1_k\rangle\right) |0_{n-k}\rangle / \sqrt{2}$. The following Proposition 3.2.1 intuitively explains this success and shows that the sequence of states $|\Psi_0\rangle = |0_n\rangle, |\Psi_1\rangle, \ldots, |\Psi_n\rangle = |GHZ_n\rangle$ gets closer and closer to the target state $|GHZ_n\rangle$, with a guaranteed improvement every two steps. The proof is in section B.6.

Proposition 3.2.1. *For any $k = 0, \ldots, n$, let $D_k = D_{EM}(|\Psi_k\rangle\langle\Psi_k|, |GHZ_n\rangle\langle GHZ_n|)$. We have $n/2 \leq D_0 \leq (n+1)/2$, $D_n = 0$, and $(n-k)/2 \leq D_k \leq \left(n - k + \sqrt{2}\right)/2$ for any*

$k = 1, \ldots, n-1$. *In particular, we have* $D_{k+2} < D_k$ *for any* $k = 0, \ldots, n-2$.

Furthermore, as shown in Figure 3-1b, optimization using the quantum EM distance is, in virtually all cases, successful at learning the GHZ state. In the simulations for Figure 3-1b, the quantum EM distance is efficiently estimated (lower bounded) using the dual formulation by considering the expectation of the generated state over a subset of $O(n)$ Hermitian operators H_i all with Lipschitz constant equal to one.

$$\begin{aligned} \tilde{D}_{EM} &= \max_{H_i} \left| \langle \psi(\boldsymbol{\theta}) | H_i | \psi(\boldsymbol{\theta}) \rangle - \langle GHZ_n | H_i | GHZ_n \rangle \right| \\ &\leq D_{EM}\left(|\psi(\boldsymbol{\theta})\rangle \langle \psi(\boldsymbol{\theta})|, |GHZ_n\rangle \langle GHZ_n| \right), \end{aligned} \tag{3.8}$$

where $\tilde{D}_{EM}$ is the approximation which lower bounds D_{EM} by taking the maximum over the $O(n)$ operators H_i chosen for optimization of the circuit (see section B.5 for list of operators). Using $\tilde{D}_{EM}$, we can successfully learn and generate the GHZ state regardless of the size of the system. Given the simplified form of our circuit, calculating $\tilde{D}_{EM}$ can be efficiently performed using a classical computer and the methodology is detailed in section B.5. Interestingly, though the subset of Hamiltonians considered in calculating $\tilde{D}_{EM}$ is significantly less than the total space of Hamiltonians available needed to exactly calculate D_{EM}, the simplified form of $\tilde{D}_{EM}$ still suffices to completely learn the GHZ state. This perhaps surprising fact is one motivation for our qWGAN, discussed later, which uses similar techniques to construct a general algorithm for learning quantum data in more complex settings.

3.2.2 Properties of quantum EM distance in learning settings

For the EM distance over probability distributions, the classic work of [23] showed that the EM distance has a number of properties that confer advantages in learning settings over other distance metrics such as the total variational distance. Here, we show that our quantum EM distance offers corresponding analogous properties when learning in quantum settings. These properties provide intuitive explanations for why learning was so successful in the toy model analyzed earlier. First, the quantum EM distance is super-additive with respect to the tensor product [108, Proposition 4]:

Proposition 3.2.2. *For any two quantum states ρ, σ of n qubits and any $k = 1, \ldots, n-1$,*

$$D_{EM}(\rho, \sigma) \geq D_{EM}(\rho_{1\ldots k}, \sigma_{1\ldots k}) + D_{EM}(\rho_{k+1\ldots n}, \sigma_{k+1\ldots n}), \tag{3.9}$$

where $\rho_{1\ldots k}$ and $\rho_{k+1\ldots n}$ are the marginal states of ρ over the first k and the last $n-k$ qudits, respectively, and analogously for σ.

In the case of product states where $\rho = \rho_{1\ldots k} \otimes \rho_{k+1\ldots n}$ and $\sigma = \sigma_{1\ldots k} \otimes \sigma_{k+1\ldots n}$, then the above is an equality:

$$D_{EM}(\rho, \sigma) = D_{EM}(\rho_{1\ldots k}, \sigma_{1\ldots k}) + D_{EM}(\rho_{k+1\ldots n}, \sigma_{k+1\ldots n}), \tag{3.10}$$

Intuitively, the Proposition above implies that operations which reduce the distance between two states over a portion of their qubits will proportionally reduce the total distance over all of the qubits. Note that no unitarily invariant distance can have this property. For example, to learn a target state $|GHZ_2\rangle |1\rangle$, updating the state $|000\rangle$ to $|GHZ_2\rangle |0\rangle$ results in a significant improvement in the quantum EM distance but, since the updated state is still orthogonal to the target state, no unitarily invariant distance will show any improvement. As an aside, super-additivity is relevant in noisy contexts as it implies that if noise only affects a small number of qubits, then the change in the EM distance is correspondingly bounded by the number of qubits on which the noise acts.

A second useful property of the quantum EM distance is that it recovers the classical earth mover's distance for quantum states diagonal in the canonical basis, and in particular, it recovers the classical Hamming distance for the quantum states of the computational basis [108, Proposition 6]:

Proposition 3.2.3. *Let p, q be probability distributions on $\{0,1\}^n$, and let*

$$\rho = \sum_{x \in \{0,1\}^n} p(x) |x\rangle\langle x|, \qquad \sigma = \sum_{y \in \{0,1\}^n} q(y) |y\rangle\langle y|. \tag{3.11}$$

Then, $D_{EM}(\rho, \sigma) = D_{EM}(p, q)$. In particular, the quantum EM distance between vectors of the canonical basis coincides with the Hamming distance: $D_{EM}(|x\rangle\langle x|, |y\rangle\langle y|) = h(x, y)$ for any x, $y \in \{0,1\}^n$.

The above proposition implies that advantages conferred in classical machine learning

algorithms when using the classical EM distance directly translate into quantum settings when using the quantum EM distance. Finally, the quantum EM distance is always contained between the trace distance and n times the trace distance [108, Proposition 2]:

Proposition 3.2.4. *For any two quantum states ρ, σ,*

$$D_1(\rho, \sigma) \leq D_{EM}(\rho, \sigma) \leq n\, D_1(\rho, \sigma)\,. \tag{3.12}$$

In particular, a small quantum EM distance guarantees that the trace distance is also small and vice-versa. Thus, convergence in the quantum EM distance necessarily implies convergence in more conventional quantum distance metrics such as fidelity or trace distance.

3.2.3 EM Distance Evaluation

As in the classical Wasserstein GAN [23], approximations to the EM distance are required to construct learning algorithms that have efficient runtimes. Note, the quantum EM distance can be exactly evaluated using algorithms for semidefinite programs [57, 317] which run in time polynomial in the dimension of the quantum state and the number of constraints. Such an exact approach would require algorithmic runtimes that are exponential in the number of qubits and furthermore, do not lead to obvious methods for calculating the gradient of the quantum EM distance. Instead, we provide a procedure below to estimate the quantum EM distance between two distributions of quantum states using its dual formulation equation 3.4. To avoid cumbersome computation of Lipschitz constants, we construct a parameterized family of functions which preserve a quantum Lipschitz constraint upon optimization. Let

$$H = \sum_{\mathcal{I} \subseteq \{1, \ldots, n\}} H_{\mathcal{I}}\,, \tag{3.13}$$

where each $H_{\mathcal{I}}$ acts non-trivially only on the qubits in the corresponding set $\mathcal{I}$. Then, Proposition 10 of [108] provides an upper bound to the quantum Lipschitz constant of a Hamiltonian in terms of its local structure.

$$\|H\|_L \leq 2 \max_{i=1, \ldots, n} \left\| \sum_{i \in \mathcal{I} \subseteq \{1, \ldots, n\}} H_{\mathcal{I}} \right\|_\infty , \tag{3.14}$$

where the maximum is taken over the qubits. The notation $i \in \mathcal{I} \subseteq \{1, \ldots, n\}$ indicates that the sum is taken only over the set of operators which act non-trivially (*i.e.,* not the identity) on qubit i. Intuitively, since the Lipschitz constant bounds the change in a Hamiltonian induced by changes to a single qubit, the bound above can be viewed as a bound on the maximum singular value of nontrivial operators thus also bounding the corresponding Lipschitz constant. A natural choice for the operators $H_{\mathcal{I}}$ are a subset of the Pauli operators which we explore in our construction of a quantum generative adversarial network next.

3.3 qWGAN Algorithm

Our quantum Wasserstein generative adversarial net (qWGAN) consists of a discriminator and generator which approximates a target distribution over states ρ_{tar} by "playing" a min-max game. Here, the generator sets its parameters θ outputting a state $G(\theta)$, and the discriminator $H(W)$ is a parameterized sum of Hermitian operators with weights W. In each iteration of optimization, the discriminator first sets its operator weights, outputting a Hamiltonian $H_{\max}$ which is the Hamiltonian maximizing our dual formulation estimate of $D_{EM}(G(\theta), \rho_{\text{tar}})$. Then, a gradient update is performed on the parameters of the generator θ. This iterative process is repeated either until convergence in the generator parameters θ or until a stopping criterion is reached. We detail the forms of the discriminator and generator as well as the steps of the algorithm in this section.

3.3.1 Form of the discriminator

In an optimal scenario, a discriminator explores the complete set of Hamiltonians which have Lipschitz constant less than or equal to one. However, this ideal case does not lend itself to efficient algorithms, and we instead construct a discriminator which efficiently estimates (lower bounds) the quantum EM distance. The discriminator we choose is a parameterized sum of strings of Pauli operators:

$$H(W) = \sum_{P_1, \ldots, P_n \in \{I,X,Y,Z\}} w_{P_1 \ldots P_n} \, \sigma_{P_1}^{(1)} \otimes \sigma_{P_2}^{(2)} \otimes \ldots \otimes \sigma_{P_n}^{(n)} , \tag{3.15}$$

where σ_I is the 2×2 identity matrix, σ_X, σ_Y and σ_Z are the Pauli matrices, superscripts specify the qubit on which the corresponding Pauli matrix acts and each $w_{P_1 \ldots P_n}$ is the

trainable parameter for the corresponding Pauli string $\sigma_{P_1}^{(1)} \otimes \sigma_{P_2}^{(2)} \otimes \ldots \otimes \sigma_{P_n}^{(n)}$. To simplify notation, we denote the set of all trainable parameters as W. Finding the exact Lipschitz constant of the Hamiltonian equation 3.15 can be computationally expensive. However, noting that Pauli operators have infinity norm of 1 and applying the triangle equality, equation 3.14 provides an easily computable upper bound to $\|H(W)\|_L$, which we denote with $\|H(W)\|_{\tilde{L}}$:

$$\|H(W)\|_{\tilde{L}} = 2 \max_{i=1,\ldots,n} \sum_{P_1,\ldots,P_n \in \{I,X,Y,Z\}: P_i \neq I} |w_{P_1 \ldots P_n}| \geq \|H(W)\|_L \,. \tag{3.16}$$

The Hamiltonian equation 3.15 has 4^n parameters and is impractical to train. For this reason, we restrict optimization to operators that contain only terms acting on few qubits. One option is to choose the set of k-local (*i.e.,* acting on k qubits and not necessarily geometrically local) Pauli operators as the discriminator. We denote with $\mathcal{O}_n^{(k)}$ the linear span of such operators. For example, the most general element of $\mathcal{O}_n^{(2)}$ is

$$H(W) = w_I + \sum_{i=1}^{n} \sum_{P \in \{X,Y,Z\}} w_P^{(i)} \, \sigma_P^{(i)} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{P,\, Q \in \{X,Y,Z\}} w_{P,Q}^{(i,j)} \, \sigma_P^{(i)} \otimes \sigma_Q^{(j)}, \tag{3.17}$$

where each w_I, $w_P^{(i)}$ and $w_{P,Q}^{(i,j)}$ is the trainable parameter for the corresponding Pauli operator. For $k \ll n$, there are $O(n^k)$ total terms in the above summation, polynomial in the number of qubits.

We can now define an approximated EM distance by restricting the optimization in equation 3.4 to k-local Hamiltonians and replacing the exact Lipschitz constant with the approximated Lipschitz constant equation 3.16:

$$D_{EM}^{(k)}(\rho, \sigma) = \max \left\{ \mathrm{Tr}\left[(\rho - \sigma)\, H\right] : H \in \mathcal{O}_n^{(k)},\ \|H\|_{\tilde{L}} \leq 1 \right\} . \tag{3.18}$$

The approximated quantum EM distance is increasing with respect to k and provides a lower bound to the exact quantum EM distance, *i.e.*, for any two quantum states ρ and σ,

$$D_{EM}^{(1)}(\rho, \sigma) \leq D_{EM}^{(2)}(\rho, \sigma) \ldots \leq D_{EM}^{(n)}(\rho, \sigma) \leq D_{EM}(\rho, \sigma) \,. \tag{3.19}$$

The order k of Pauli operators can be tuned to the complexity of a given problem. To

learn truly random states, access to the full set of Hamiltonian operators is needed, but in many cases, especially when learning data generated by shallow circuits or data that can be discriminated via reduced density matrices, learning using only lower order Pauli operators can be effective.

The approximated EM distance equation 3.18 can be computed with the following linear program, which can be efficiently solved. To simplify notation, we assume all parameters are enumerated in a list $W = \{w_1, w_2, \ldots, w_{|W|}\}$. For each parameter w_i, we let $\mathcal{I}_i$ be equal to the set of qubits which the corresponding Pauli string acts on. Thus, with $|W|$ parameters and n qubits, one maximizes the following linear program:

$$\begin{aligned} &\text{maximize} \quad \sum_{j=1}^{|W|} c_j w_j \\ &\text{subject to} \quad \sum_{j:i\in\mathcal{I}_j} |w_j| \leq 1, \qquad i = 1, ..., n \end{aligned} \tag{3.20}$$

where c_j is the trace of the product between the j-th Pauli string and $G(\theta) - \rho_{\text{tar}}$. *i.e.,* assuming w_j is associated to Pauli string $\sigma_{P_a}^{(a)}\sigma_{P_b}^{(b)}\ldots\sigma_{P_k}^{(k)}$, then $c_j = \text{Tr}\left[(G - \rho_{\text{tar}})\sigma_{P_a}^{(a)}\,\sigma_{P_b}^{(b)}\ldots\sigma_{P_k}^{(k)}\right]$. In the above formulation, there exists a constraint for each qubit i limiting the sum of magnitudes of operators acting on that qubit to less than or equal to one.

The linear program in equation 3.20 can be transformed into a standard form linear program with n constraints (one for each qubit), which outputs a sparse set of at most n non-zero weights [47] (the number of non-zero variables in linear programs in standard form is at most the number of constraints). Specifically, the linear program will output $n_{\text{active}} \leq n$ operators with non-zero weights, called active operators, constructing a Hamiltonian $H_{\max}$ which is passed onto the generator for optimization:

$$H_{\max} = \sum_{i=0}^{n_{\text{active}}} w_i^* H_i^*, \tag{3.21}$$

where w_i^* and H_i^* are the weights and active operators respectively. As an example, we show in section B.8 that for product states, the linear program outputs an optimal Hamiltonian composed of single qubit Pauli terms chosen by selecting the single qubit Pauli term which has the greatest contribution on each qubit (thus $n_{active} = n$ in this setting). Since $n_{\text{active}} \leq n$, gradient updates on the generator which aims to minimize the expectation of $H_{\max}$ can

be performed. However, this efficiency comes at the cost of potentially missing certain active operators in the optimal Hamiltonian. Equation 3.21 is naturally biased towards including low order operators in its set of active operators. Especially for problems where the optimization needs to access higher order operators, this methodology may fail to perform effectively.

section B.8 compares values of the estimated distance $D_{EM}^{(2)}$ with the actual distance D_{EM} for random states generated by shallow circuits showing that there is a correlation between the two measures, although the approximation may introduce an unwanted bias. As we show in section B.7, restricting the optimization over operators to terms acting on few qubits does not affect the value of the distance, since the optimal unconstrained Hamiltonian for the maximization problem equation 3.20 contains only these terms whenever the coefficients c_j associated to single Pauli operators are all $\Omega(1)$. Furthermore, since operators only act on few qubits, efficient algorithms from [176, 177] can be applied to calculate the expectations of $|W|$ Pauli operators in equation 3.17 using $O(\log |W|)$ copies of the generated and target states. If a large number of higher order Pauli terms are included in the decomposition above, such logarithmic dependence may no longer hold.

We stress that the approximation employed here is the central limitation in successfully applying our qWGAN model. Improving this approximation is crucial to expanding the scope of application of our qWGAN beyond the assorted examples provided in section 3.4. Generally, there are two paths by which one can improve the approximation. The first path consists of identifying a more optimal relaxation to the semidefinite program in comparison to the linear relaxation of Equation 3.20. Existing quantum algorithms for solving semidefinite programs [57, 317] may help in realizing this goal. The second path consists of finding methods to more optimally choose the subset of Hamiltonians or Pauli operators included in the approximation. In some cases, higher order Pauli operators are needed to distinguish states and avoiding these altogether may produce sub-optimal results. In light of this, we describe below one technique that empirically helps in finding a good subset of Pauli operators.

Optional cycling of operators Over a small number of steps of optimization, changes to the expectations of operators c_j are expected to be very small. Therefore, if the expectation of a given operator in the discriminator is small, it is unlikely that the operator will be

chosen as an active operator over the course of optimization. Therefore, one has the option of removing these “bad" operators and including new operators (here, we choose the new operators uniformly randomly from the set of all Pauli operators) into the set of operators over which the discriminator optimizes. Many choices exist for cycling operators; here, we opt for a simple choice where operators are cycled out when the expectation of an operator is below a threshold equal to $P(\min_i c_i^*)$ (*i.e.,* minimum taken over all active operators) where $0 < P \leq 1$. When an operator is cycled out, a random Pauli operator is then included in the discriminator's set of operators including potentially Pauli operators that were removed in earlier cycles. Cycling operators may, of course, be detrimental if operators are cycled out that end up being useful during later phases of training. Nevertheless, in our experiments, we often find that the amount of cycling can serve as another tunable hyperparameter for improving the performance of our qWGAN.

3.3.2 Form of the generator

In its most general form, a generator is an object or function, that when given an input (potentially a sample from a random variable), outputs a state which approximates or produces a sample drawn from a distribution close to the target distribution. Similar to classical machine learning where neural networks are customized to given settings – *e.g.,* convolutional neural networks optimized for image analysis [215, 153, 221] and transformer networks optimized for text analysis [319, 59, 113] – the form of the generator in our quantum algorithm can and should be customized to the specific problem setting. Many options exist for constructing a generator including parameterized quantum circuits [43, 119] and quantum neural networks [298, 295, 209, 98]. The form of the generator determines the space of functions which a generator can access, and ideally this space should overlap with the function of the target object. Given we can only cover a limited class of generators in our analysis, we focus here on a single, though generic, form for the generator, encouraging future research to construct and analyze generators customized to specific applications in quantum machine learning.

In this generic formulation, the generator $G(\theta)$ is a function which maps a starting state $|\psi_0\rangle \langle\psi_0|$ to a density matrix ρ representing the distribution over quantum states that one aims to reconstruct. As in [79], our generator is constructed by a set of probabilities and

associated parameterized unitaries $\{(p_1, U_1), \ldots, (p_r, U_r)\}$:

$$G(\theta) = \sum_{i=1}^{r} p_i U_i \left|\psi_0\right\rangle \left\langle\psi_0\right| U_i^\dagger, \tag{3.22}$$

where we use θ to denote the set of all parameters for the generator which includes the probabilities p_i and parameters for each unitary U_i. r is the maximum rank of the output density matrix which can be tuned as a hyperparameter. Later, we consider U_i constructed by parameterized quantum circuits with one and two qubit gates. The choice of these parameterized circuits depends on the nature of the problem (see section 3.4 for examples).

3.3.3 qWGAN optimization procedure

The algorithm for the qWGAN detailed in algorithm 1 iteratively optimizes parameters of the generator and discriminator, consistent with methods used in classical GANs [23]. The following two steps are repeated until convergence in the parameters of the generator θ. First, the parameters w are updated using the linear program equation 3.20 to maximize the quantum EM distance D_{EM} in equation equation 3.4. Then, a gradient update is performed on the parameters of the generator θ.

Algorithm 1 qWGAN with quantum earth mover's distance

Require: initial discriminator operators: $H_i^{[0]}$ ▷ *e.g.,* set of 2-local Paulis
Require: initialization of generator parameters: $p_i^{[0]}$ and $\theta_i^{[0]}$
Require: hyperparameters for generator optimizer (*e.g.,* learning rate α)
1: **while** θ, p have not converged **do** ▷ alternatively, stop after T steps
discriminator optimization:
2: measure operator expectations: $c_i \leftarrow \mathrm{Tr}\left[H_i(G(\theta) - \rho_{\mathrm{tar}})\right]$
3: find w_i^*, H_i^* (linear program, equation equation 3.20) ▷ $H_{\max} = \sum_i w_i^* H_i^*$
4: **optional:** cycle operators
generator optimization:
5: find gradients g_p, g_θ of $\mathrm{Tr}\left[G(\theta) H_{\max}\right]$ ▷ see section B.9
6: perform gradient update on θ and p ▷ *e.g.,* $\theta \leftarrow \theta - \alpha g_\theta$

3.3.4 Properties of the gradient

As stated earlier, one can calculate the gradients with respect to the parameters of the unitary operator implemented by the circuit (step 5 in algorithm 1) via the parameter shift rule [293]. As an example, let $G(\theta)$ be generated by the quantum circuit that implements

the unitary operator

$$U(\theta) = \prod_k U_k \, e^{-i\theta_k P_k} \,, \tag{3.23}$$

where each U_k is a unitary operator that does not contain any parameter and each P_k is a generalized Pauli operator. Then, given an optimal Hamiltonian $H_{\max}$, one can calculate the gradient as follows for a given parameter θ_k:

$$\frac{\partial}{\partial \theta_k} \operatorname{Tr}\left[G(\theta) H_{\max}\right] = \operatorname{Tr}\left[G(\theta^+) H_{\max}\right] - \operatorname{Tr}\left[G(\theta^-) H_{\max}\right], \tag{3.24}$$

where θ^+ and θ^- are the values of the parameters shifted by $\frac{\pi}{4}$ in either direction for the entry corresponding to θ_k [293]. This parameter-shift rule has the benefit that the equation for the gradient is in fact exact (not an approximation). Gradients for each individual parameter must be calculated using separate circuit evaluations.

Prior work has shown that local cost functions avoid barren plateaus up to poly-logarithmic depth in the circuit [78]. Due to the super-additivity property of the quantum EM distance and the construction of the linear relaxation in equation 3.20, the optimal Hamiltonian is heavily biased towards local terms and thus fits into this regime. We formalize this below by showing that high order Pauli strings acting nontrivially on k qubits must have magnitude greater than k times the smallest single qubit Pauli contribution to the optimal Hamiltonian $H_{\max}$.

Proposition 3.3.1. *Let $w^* : \{I, X, Y, Z\}^n \to \mathbb{R}$ be the set of parameters that achieve the maximum in equation 3.20, and let*

$$a = \min_{i=1,\ldots,n} \max_{P=X,Y,Z} \left| \operatorname{Tr}\left[\left(G(\theta) - \rho_{\text{tar}}\right) \sigma_P^{(i)} \right] \right| \,. \tag{3.25}$$

Then, $w^*_{P_1 \ldots P_n} = 0$ for any $P_1, \ldots, P_n \in \{I, X, Y, Z\}$ such that

$$|c_{P_1 \ldots P_n}| < a \, |\{i = 1, \ldots, n : P_i \neq I\}| \,. \tag{3.26}$$

In particular, $w^*_{P_1 \ldots P_n} = 0$ for any Pauli string that acts nontrivially on more than $2/a$ qubits.

The proof of the above is deferred to section B.7. As a corollary of the above, any global k-qubit Pauli term that appears in the optimal Hamiltonian must exceed the sums of

the maximum single qubit Pauli terms on the k qubits on which it acts. Thus, the optimal Hamiltonian $H_{\max}$ will be at most $2/a$-local which is constant when the expectation of single qubit Paulis is $\Omega(1)$. In these settings, the qWGAN will avoid barren plateaus whenever the generator has depth that does not grow faster than logarithmically in the number of qubits (see Theorem 2 of [78]).

3.4 Experiments

Here, we provide examples of settings where the qWGAN is effective at learning quantum data. We apply the qWGAN directly to the toy model of subsection 3.2.1 and also consider a more general scenario where the qWGAN learns states generated by a mixing circuit previously known to suffer from barren plateaus [246, 78, 180]. In section B.10, we include results for the qWGAN in two other scenarios: one where the generator is a circuit formed by a quantum alternating operator ansatz (QAOA) [130, 157, 128] and one where the qWGAN is tasked with learning mixed states. The Adam optimizer with a default learning rate of 0.01 was used to train the qWGAN [212]. Details on the structure of the quantum circuits and on how the simulations were performed are provided in section B.11 and section B.12, respectively.

Learning the GHZ state The n-qubit GHZ state is an entangled state which requires a simple circuit of depth n to construct. However, as noted in subsection 3.2.1, the correct parameters of this circuit are hard to learn when using cost metrics that are a function of the inner product between the generated and target GHZ state. Continuing our analysis, we show that our qWGAN is especially efficient and effective at learning the correct parameters of a circuit to generate the GHZ state.

As shown in Figure 3-2, our qWGAN efficiently generates the $n = 8$ qubit GHZ state in around 500 steps of optimization– results for circuits of different size are also consistent with this analysis and detailed in section B.10. The generator circuit has $n + 2$ parameters and depth n. Simulations are repeated 50 times across random initializations. The discriminator starts with access to $k = 2$ local Pauli operators, cycling out "bad" operators every five steps. Estimated EM loss is also plotted in Figure 3-2 (normalized by dividing by the number of qubits), which is equal to the quantum EM distance as measured by the active operators in the discriminator (lower bounding the actual quantum EM distance). Jumps

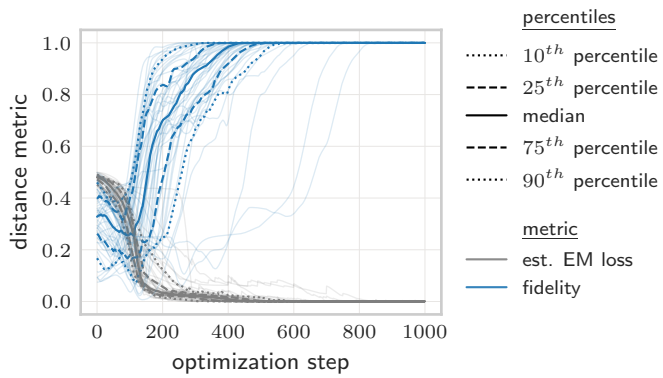
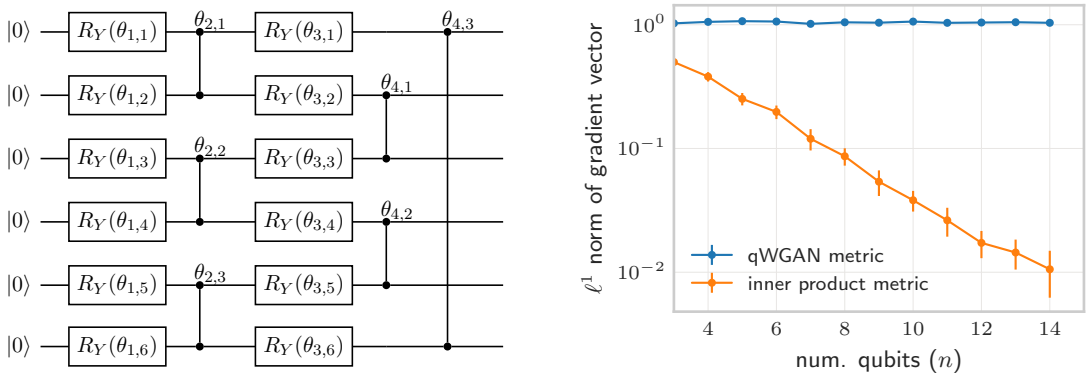


Figure 3-2: The qWGAN consistently generates the 8 qubit GHZ state. Estimated EM loss (quantum EM distance estimated by active operators) is plotted in grey alongside the fidelity in blue. EM distance is normalized to a maximum of one by dividing by the number of qubits. Percentiles are calculated across 50 simulations. Individual simulations are plotted as transparent lines.

in the estimated EM loss can be observed when operators are cycled and later become active, highlighting the importance of randomly cycling operators in these simulations. It is interesting to note that during the early phases of learning, the qWGAN often optimizes the EM distance while temporarily decreasing the fidelity. This learning profile is typically associated with transitions from the state $|0_n\rangle$ to the GHZ state. As our toy model indicated, this transition characterized by a temporary decrease in the fidelity is needed to reach the global optimum.

Teacher-student learning To analyze our qWGAN in a more general setting, we consider a "teacher-student" setup where the circuit used to generate the target state and perform learning are both of the form shown in Figure 3-3a. This circuit is a generic mixing circuit also studied in [246, 78, 180] where barren plateaus in the loss landscape are observed. For our simulations, the target state ϕ_{tar} is generated by a depth 2 circuit (*i.e.* gates shown in Figure 3-3a repeated twice) with parameters drawn i.i.d. from the standard normal distribution. As a point of comparison, we compare our qWGAN to a quantum GAN equipped with the loss function $F = 1 - |\langle\phi_{\text{tar}}|\phi(\boldsymbol{\theta})\rangle|^2$ which is a function of the inner product between the target and generated state. Figure 3-3b shows that when a circuit of the same form is used to learn the target state, gradients of F (function of the inner product) decay exponentially with more qubits whereas gradients of the quantum EM loss function remain constant. Note that the exponentially decaying gradients for the inner

product loss metrics are observed here for constant depth shallow circuits. This result further confirms that loss landscapes for the quantum EM distance avoid common pitfalls faced by conventional distance metrics including the Wasserstein semi-metric proposed in [79].



(a) (b)

Figure 3-3: (a) Single layer of mixing circuit consisting of alternating layers of parameterized Pauli Y rotations and parameterized Pauli Z-Z rotations applied to pairwise qubits. Here, the form of the circuit is shown for six qubits. (b) Learning using two layers of mixing circuit (shallow, constant depth) results in exponentially decaying gradients for conventional loss metrics. Gradients of the qWGAN remain constant while gradients with respect to a loss metric as a function of the inner product decay exponentially in the number of qubits. Gradients are calculated at first step of optimization and ℓ^1 norm is divided by n to normalize to the number of parameters in the circuit. Findings are consistent when the average is taken for the ℓ^2 norm or of individual gradient entries as shown in section B.10. Results are averaged across 100 simulations for each data point.

Furthermore, as shown in Figure 3-4, the qWGAN successfully learns the states constructed by $n = 8$ qubit teacher circuits using student circuits of depth 4. This circuit has 96 trainable parameters. Simulations are repeated 50 times across random initializations in Figure 3-4. Target states are generated by drawing the parameters of the teacher circuit i.i.d. from the standard normal distribution. Learning is typically achieved within a few hundred steps of optimization. In these simulations, the discriminator for the qWGAN contains all order 2 Pauli operators and no cycling of the operators was performed. We find that, in general, learning in the teacher-student setting is best achieved when the student circuit is deeper than the teacher circuit. Furthermore, due to the approximations made in calculating the quantum EM distance, we find that our algorithm struggles to learn especially deep 8-qubit teacher circuits which are four layers or more in depth. For these deeper target

circuits, we suspect that higher order Paulis are needed to efficiently estimate the quantum EM distance, and further improvements to the optimization must be made to incorporate these higher order Paulis in the estimation procedure. Additional simulations for different circuit sizes are shown in section B.10.

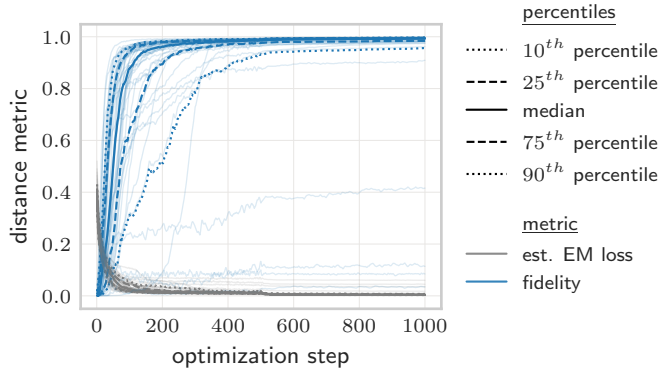


Figure 3-4: The student circuit is able to approximate well the state generated by the teacher circuit. Here, the target is constructed by randomly setting the parameters of a depth 2 mixing circuit (teacher circuit). The qWGAN, equipped with a depth 4 generator circuit, successfully learns the target state by optimizing over the quantum EM distance. Estimated EM loss (quantum EM distance estimated by active operators) is plotted in grey alongside the fidelity in blue. EM distance is normalized to a maximum of one by dividing by the number of qubits. Percentiles are calculated across 50 simulations. Individual simulations are plotted as transparent lines.

3.5 Discussion

As interest in quantum machine learning algorithms has flourished, recent research has highlighted the challenges associated with learning using quantum computers. At the root of these challenges are adverse properties of loss landscapes in quantum machine learning settings, perhaps most notably that loss landscapes have poor local minima and exponentially decaying gradients. In this work, we show that the loss landscape induced by the quantum EM distance can potentially confer advantages in machine learning settings, especially when optimization is performed over local gates in shallow circuits. Our results provide a new approach to constructing loss landscapes which can avoid common quantum machine learning roadblocks.

For the specific application of learning quantum data, we have proposed a qWGAN which leverages the quantum EM distance to produce an efficient learning algorithm. In

accord with its classical counterpart [23], we show that our qWGAN can potentially improve convergence and stability in learning quantum data. Nevertheless, the qWGAN struggles in learning more “random" data or data generated from deep circuits. These challenges stem from two approximations made in the learning procedure. First, to ensure runtime efficiency, the discriminator was restricted to measuring a subset of local Pauli operators. Second, the optimization statement to calculate the quantum EM distance was relaxed into a linear program to ensure it can be efficiently calculated classically. Though this estimated distance is well correlated to the true distance when learning certain structured states like the GHZ state or states generated by shallow circuits, it fails to tightly bound the quantum EM distance in more challenging settings. Looking forward, improving the bounds given by relaxations of the quantum EM distance can potentially allow for application of our qWGAN in these more challenging settings.

Beyond the test cases studied here, the qWGAN has many potential applications. In quantum controls, one can use it to search for robust or optimal control parameters [138, 266]. For unsupervised learning, the qWGAN provides a framework and approach to quantum circuit compression, data encoding, and sampling [294, 286, 188]. For quantum error correction, one can use a qWGAN to develop new techniques for constructing quantum error codes or assisting error correction procedures [257, 33, 37, 187].

Chapter 4

Efficient classical algorithms for simulating symmetric quantum systems

Quantum algorithms in general struggle with a version of the curse of dimensionality associated to the exponentially growing 2^n dimensional vector space of n qubits. Systems constrained under symmetry live in subspaces of dimension less than 2^n that can grow much slowly, perhaps even polynomially with n. This raises the question of whether quantum advantage may exist in such situations where the quantum system we aim to learn is constrained under symmetries. This work studies that question showing that the answer is not so simple; if the system is constrained too much, then there may also exist classical algorithms that can perform learning in polynomial time.

4.1 Introduction

In the physical sciences, symmetries are useful for simplifying difficult computational tasks by reducing the effective degrees of freedom of the problem. This general principle has been used to find exact solutions to many problems, such as integrable systems [48], topological fixed-point models [222], or conformal field theories [39]. There has been a hope that similar symmetries may enable the efficiency of quantum algorithms for simulating or finding the ground state of a symmetric Hamiltonian. Indeed, it is known that there exist theoretical

guarantees for quantum algorithms for finding the ground state [292] and fast-forwarding quantum dynamics [154] of Hamiltonians which commute under the action of the symmetric group S_n on qubits. It has also numerically been shown that quantum algorithms are capable of finding the ground state of certain integrable systems [335, 20] even when the symmetry is not explicitly given to the quantum algorithm *a priori*. Furthermore, prior work used Lie algebraic methods to efficiently classically simulate operators restricted to a Lie algebra whose dimension is polynomially large (independent of the potentially exponentially large Hilbert space dimension) [307, 349]. Quantum machine learning models that are symmetry equivariant are also believed to be more efficiently trainable than their general counterparts [344, 19, 73, 247, 218, 282]. These quantum models are partly inspired by classical neural network models that have enjoyed much recent success [58, 338, 94]. However, restricting quantum algorithms to problems obeying many symmetries potentially allows for efficient classical algorithms which also take advantage of these same symmetries. This raises the natural question: are there efficient classical algorithms capable of performing these tasks?

This is what we investigate here. Intuitively, we show that problems constrained by large symmetry groups yield efficient classical algorithms for computing many properties of interest, as illustrated in Fig. 4-1(a). We first give a very general classical algorithm for finding the ground state and energy of Hamiltonians constrained by many symmetries. We also consider the problem of simulating dynamics under symmetric Hamiltonians. We then specialize to the case of systems invariant under permutations of its qubits. Finally, we dequantize an algorithm for performing binary classification problems using permutation-invariant systems on qubits.

4.2 Motivation and setting

Our algorithms are motivated by the fact that symmetries significantly reduce the number of degrees of freedom for a given problem. For example, consider the classical setting of boolean functions which are invariant under arbitrary permutations of the bits. Such functions are defined up to the orbits of the boolean cube with respect to permutations of the bits. For a boolean function on n bits, there are $n+1$ orbits indexed by the Hamming weight of the bitstrings. Therefore, any problem over symmetric boolean functions need only consider

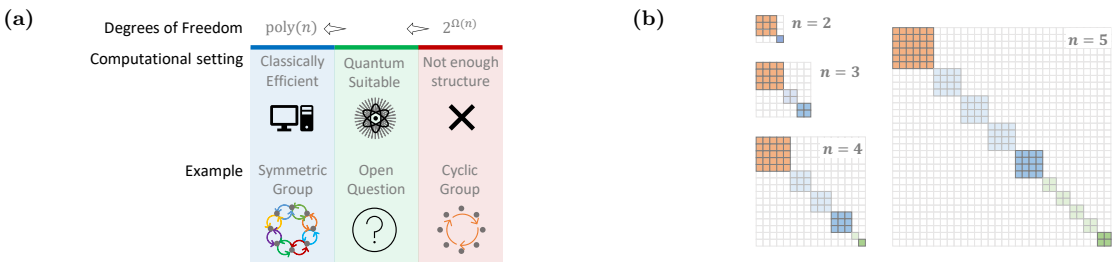


Figure 4-1: (a) Small groups of symmetry leave too large of an effective dimension for the problem to be tractable via quantum computation. On the contrary, very restrictive symmetries render a problem classically tractable. Between these two regions lies an area of promise where quantum computers may offer an advantage. (b) The Schur–Weyl decomposition shows that only a smaller representative subspace (indicated by darker colors) of the larger Hilbert space needs to be considered for permutation invariant operations. The size of this subspace grows as $\mathrm{O}(n^3)$ for n qubits.

a given element of each of the $n+1$ orbits to cover all possible degrees of freedom. As we will later show, the symmetric group acting over n qubits similarly reduces systems to $\mathrm{O}(n^3)$ degrees of freedom. By considering the algebra of the symmetric group on the symmetric subspace of linear operators, we will show that all these degrees of freedom can be manipulated solely through classical computation.

Before proceeding, we need to introduce important functions and definitions that will be used in this setting. We first formalize the notion of symmetry by speaking of *invariant operators*, defined in the following way:

Definition 4.2.1 (Invariant operator)**.** Given a compact group G with unitary representation $R : G \to U(N)$, a linear operator $H : \mathbb{C}^N \to \mathbb{C}^N$ is invariant under $R(G)$ if

$$R(g)HR(g)^\dagger = H, \quad \forall g \in G. \tag{4.1}$$

Note that any invariant operator is also an *equivariant operator* [247] in the sense that it commutes with the representation of the group.

Any operator can be projected onto the symmetric subspace induced by $R(G)$ using the twirling superoperator Re_R (more commonly known as the Reynold's operator in invariant theory) [263, 310], which maps any operator onto the set of equivariant operators:

$$\mathrm{Re}_R(M) = \frac{1}{|G|}\sum_{g\in G} R(g)MR(g)^\dagger. \tag{4.2}$$

Invariant subspaces of a larger Hilbert space can be identified by performing an isotypic decomposition of the representation of a group. As an example, in the case of systems invariant under permutations of the qudits, the Schur decomposition maps the computational basis into blocks of invariant subspaces. We graphically visualize this phenomenon in Fig. 4-1(b) and provide further details in the Supplementary Information.

Throughout this study, runtime complexities are denoted as a function of the matrix multiplication exponent ω. The best known upper bound is currently $\omega = 2.37188$ [122], which implies asymptotic runtimes of $\mathrm{O}(n^{\omega+\alpha})$ for any $\alpha > 0$ for stably performing common linear algebraic routines including eigendecomposition, SVD, and matrix inversion [111].

4.3 Algorithms for general symmetry groups

In this Section, we discuss the general problem of finding the ground state energy, ground state, and performing time evolution under a Hamiltonian H on a finite-dimensional Hilbert space which is invariant under some representation R of a symmetry group G. Consider the *-subalgebra of operators invariant under R to which H belongs. We think of this subalgebra as a standalone *-algebra X, such that the embedding of X into the full operator algebra defines a representation A of X.

The practical relevance of these considerations is when the size of the total Hilbert space grows exponentially with some scaling parameter n. The paradigmatic example is the Hilbert space of n qubits. If there are enough symmetries, it can happen that the dimension $N(n)$ of X only grows polynomially with n, in which case many properties can be calculated efficiently [307]. This restriction of X to a lower-dimensional subspace may more generally happen beyond systems symmetric in the sense of Definition 4.2.1. Due to this, for now we focus explicitly on X and A, rather than on G and its representation R; we will discuss the connection of our results to G and R more specifically at the end of this Section.

For the various algorithms we now consider, we will assume that different properties of X and A are known. For the algorithm for finding the ground state energy of H given in Theorem 4.3.1, we will assume that the structure constants of X in some preferred basis are known. In a slight abuse of notation, we will refer to those structure constants as $X_k^{i,j}$, where i, j, and k label basis elements. We note that the structure constants can frequently be efficiently obtained from the generators of an algebra, for example in the case of Lie

subalgebras [307, 193].

In the course of proving Theorem 4.3.2, we give an algorithm for finding the ground state of H. Every finite-dimensional *-algebra is isomorphic to a direct sum of irreducible blocks, and every representation is isomorphic to a direct sum of irreducible representations. That is, there is a block-diagonal orthonormal basis $|\lambda, q_\lambda, p_\lambda\rangle$ of the vector space acted upon by A, where λ labels an irrep of X, q_λ labels a basis vector internal to λ, and p_λ labels a basis vector in the multiplicity vector space of λ; this is the basis in which we compute the ground state of H (for some arbitrary and fixed dimension label $p_{\lambda 0}$). To prove our theorem, we assume knowledge of the matrix elements:

$$F^{i,\lambda}_{q_\lambda, q'_\lambda} \coloneqq \langle \lambda, q_\lambda, p_{\lambda 0}| \, A_i \, |\lambda, q'_\lambda, p_{\lambda 0}\rangle \, . \tag{4.3}$$

Finally, for Theorem 4.3.3, we assume the knowledge of a *symmetric transform operator* implementable on a quantum computer, i.e. an isometry V_{STO} such that:

$$V_{STO} \, |\lambda, q_\lambda, p_\lambda\rangle = |\boldsymbol{\lambda}\rangle \, |\boldsymbol{q_\lambda}\rangle \, |\boldsymbol{p_\lambda}\rangle \, . \tag{4.4}$$

Here, $|\boldsymbol{\lambda}\rangle, |\boldsymbol{q_\lambda}\rangle, |\boldsymbol{p_\lambda}\rangle$ are bitstring encodings of $\lambda, q_\lambda, p_\lambda$, respectively, in the computational basis, labeling the $\boldsymbol{\lambda}$ register, $\boldsymbol{q}$ register, and $\boldsymbol{p}$ register, respectively. An example of such an operator is the *Schur transform* [32, 31], described in more detail in the Supplementary Information.

In all three theorems, we assume we are given the Hamiltonian $H \in A(X)$ as $h \in X$ expressed in the preferred basis, such that

$$H = \sum_i h_i A_i \, . \tag{4.5}$$

We now state our main results. First, we give a simple construction of a classical algorithm for finding the ground state energy of some representation of a Hamiltonian obeying the given symmetries.

Theorem 4.3.1 (Finding the ground state energy of symmetric Hamiltonians)**.** *Consider a subalgebra X of dimension N, and assume that the structure constants of X in some preferred basis are known as discussed above. Let $H \in A(X)$ be a Hamiltonian given in the preferred basis as in Eq. equation 4.5. Then the ground state energy of H can be found in*

time $\mathrm{O}\left(N^{\omega}\right)$.

Proof. Consider the operator with indices:

$$\hat{h}_k^j \coloneqq \sum_i h_i X_k^{i,j} \,, \tag{4.6}$$

which is nothing but the regular representation of h for the algebra X. Then we have that their ground state energies are equal:

$$\mathrm{GSE}(H) = \mathrm{GSE}(\hat{h}) \,. \tag{4.7}$$

This is because the regular representation is faithful, and the ground state energy of an operator is the same in any faithful representation. Since X has dimension N, the ground state energy of $\hat{h}$ can be found in time $\mathrm{O}(N^{\omega})$. □

An advantage of this algorithm is that the only necessary information are the structure constants of X; no knowledge of the irreps of X is needed. However, due to this we have poor scaling with the number of irreps n_λ, as the direct sum structure of X is not necessarily known. Another disadvantage of this approach is that it only gives the ground state energy, rather than the ground state itself (in a representation that is not the regular representation).

We now focus on the case when we are interested in finding the ground state of some representation of such a Hamiltonian, in a basis where the action of the representation is known.

Theorem 4.3.2 (Finding the ground state of symmetric Hamiltonians)**.** *Consider a subalgebra $A(X)$, and assume that the matrix elements $F^{i,\lambda}_{q_\lambda,q'_\lambda}$ are known as discussed above. Then the ground state energy and ground state of H in the $|\lambda, q_\lambda, p_{\lambda 0}\rangle$ basis can be found in time $\mathrm{O}\left(n_\lambda n_q^{\omega}\right)$, where n_λ are the number of irreps of X and n_q the maximum irrep dimension.*

Proof. For each λ, consider the operator with indices:

$$\hat{h}^{\lambda}_{q_\lambda,q'_\lambda} \coloneqq \sum_i h_i F^{i,\lambda}_{q_\lambda,q'_\lambda} \,. \tag{4.8}$$

Note that in the $|\lambda, q_\lambda, p_\lambda\rangle$ basis, H has a block diagonal form. Furthermore, as p_λ labels isomorphic copies of irreps, we can find the ground state by fixing $p_{\lambda 0}$ WLOG. Namely, the

ground state energy is given by:

$$\mathrm{GSE}(H) = \min_{\lambda} \mathrm{GSE}(\hat{h}^{\lambda}) \,, \tag{4.9}$$

where:

$$\mathrm{GSE}(\hat{h}^{\lambda}) \coloneqq \min_{|\psi\rangle} \langle\psi| \, \hat{h}^{\lambda} \, |\psi\rangle \,. \tag{4.10}$$

Furthermore, let

$$\lambda_{\min} \coloneqq \mathrm{argmin}_{\lambda} \, \mathrm{GSE}(\hat{h}^{\lambda}) \tag{4.11}$$

and

$$|\psi^*\rangle \coloneqq \mathrm{argmin}_{|\psi\rangle} \, \langle\psi| \, \hat{h}^{\lambda_{\min}} \, |\psi\rangle \ . \tag{4.12}$$

Then, for any p,

$$|\lambda_{\min}, \psi^*, p\rangle \tag{4.13}$$

is a ground state in the $|\lambda, q_\lambda, p_\lambda\rangle$ basis. The dimension of $\hat{h}^{\lambda}$ is $\dim_X(\lambda) \times \dim_X(\lambda)$, and thus calculating $|\psi^*\rangle$ will take time $\mathrm{O}\left(\dim_X(\lambda)^{\omega}\right)$. In total, finding the ground state of H in the $|\lambda, q_\lambda, p_{\lambda 0}\rangle$ basis takes time $\mathrm{O}\left(n_\lambda \dim_X(\lambda)^{\omega}\right) = \mathrm{O}\left(n_\lambda n_q^{\omega}\right)$. □

We now show that the dynamics of an initial state under equivariant unitaries can be classically simulated even if $\rho \neq A(X)$. The given procedure is fully classical if the initial state is given as a classical shadows description of the state; if the input is given as a quantum state, we show that performing classical shadow measurements is efficient and then reduces the algorithm to the purely classical setting. This generalizes a similar approach taken in [238] in the case of particle number symmetry.

Theorem 4.3.3 (Simulating equivariant dynamics)**.** *Let*

$$O = \sum_i o_i A_i \tag{4.14}$$

be a projective measurement and

$$U = \sum_i u_i A_i \tag{4.15}$$

a unitary operator. Assume the matrix elements $F^{i,\lambda}_{q_\lambda, q'_\lambda}$ *as described previously are known.*

Assume also the existence of a symmetry transform operator V_{STO} with depth v. Then,

$$\ell(\rho) = \mathrm{tr}\left(OU\rho U^\dagger\right) \tag{4.16}$$

can be estimated to additive error ϵ with probability $1-\delta$ via $\tilde{\mathrm{O}}\left(|O|_\infty^2 \epsilon^{-2} n_\lambda^2 n_q^2 \log\left(\delta^{-1}\right)\right)$ calls to a quantum computer each of depth $v+1$, up to an additional time $\mathrm{O}\left(n_\lambda n_q^\omega\right)$ in classical processing. Here, n_λ are the number of irreps of X and n_q the maximum irrep dimension.

Proof. Let $\tilde{O}_{\boldsymbol{p_0}}, \tilde{U}_{\boldsymbol{p_0}}$ be projections of $V_{STO} O V_{STO}^\dagger, V_{STO} U V_{STO}^\dagger$ onto some particular $\boldsymbol{p}$. Classically, we can calculate:

$$M_{\boldsymbol{p_0}} = \tilde{U}_{\boldsymbol{p_0}}^\dagger \tilde{O}_{\boldsymbol{p_0}} \tilde{U}_{\boldsymbol{p_0}} \tag{4.17}$$

in time $\mathrm{O}\left(n_\lambda n_q^\omega\right)$, as it is given by the matrix multiplication of n_λ blocks each of size at most $n_q \times n_q$. $\tilde{\mathrm{O}}\left(|O|_\infty^2 \epsilon^{-2} n_\lambda^2 n_q^2 \log\left(\delta^{-1}\right)\right)$ random Pauli measurements of the state

$$\tilde{\rho} = \mathrm{tr}_{\boldsymbol{p}}\left(V_{STO} \rho V_{STO}^\dagger\right) \tag{4.18}$$

then suffice to estimate the expectation:

$$\tilde{\ell}(\rho) = \mathrm{tr}\left(M_{\boldsymbol{p_0}} \tilde{\rho}\right) \tag{4.19}$$

to additive error ϵ with probability at least $1-\delta$ using classical shadows [176]. Finally, observe that:

$$\begin{aligned} \ell(\rho) &= \mathrm{tr}\left(OU\rho U^\dagger\right) \\ &= \mathrm{tr}\left(\tilde{O}_{\boldsymbol{p_0}} \tilde{U}_{\boldsymbol{p_0}} \, \mathrm{tr}_{\boldsymbol{p}}\left(V_{STO} \rho V_{STO}^\dagger\right) \tilde{U}_{\boldsymbol{p_0}}^\dagger\right) \\ &= \tilde{\ell}(\rho). \end{aligned} \tag{4.20}$$

□

Note that in principle, the sample complexity of this procedure can potentially be improved to $\tilde{\mathrm{O}}(|O|_\infty^2 \epsilon^{-2} n_\lambda n_q^2 \log\left(\delta^{-1}\right))$ as $\tilde{\rho}$ only has $n_\lambda n_q^2$ degrees of freedom. However, the block diagonal structure over irreps is lost when transforming to the bitstring encoding $|\boldsymbol{\lambda}\rangle |\boldsymbol{q}\rangle |\boldsymbol{p}\rangle$ via V_{STO}, and thus we arrive at the sample complexity given.

In the above considerations, the group G and representation R do not directly enter. In

practice, however, we might want to start with those two. The irreps of X are in one-to-one correspondence with those of R. By a simple corollary of the von Neumann bicommutant theorem, multiplicities of irreps in X are the dimensions of the irreps of G, and the dimensions of the irreps of G are the multiplicities of the irreps of A. We thus have that:

$$\dim(X) = \sum_{\lambda} \dim_X(\lambda)^2 = \sum_{\lambda} \mathrm{mult}_R(\lambda)^2 \,. \tag{4.21}$$

Thus, the problems discussed above become classically tractable if the number n_λ of irreps of G with non-zero multiplicity in R, as well as the maximum multiplicity n_q of an irrep λ in R, are both polynomially small.

4.4 Permutation invariance on qubits

We now discuss such an example of a symmetry group with low-multiplicity irreps. Namely, we will apply the previously described procedures to the case where G is given by S_n and R is the representation on n qubits acting by permutations,

$$R(\pi)\left|i_1\right\rangle \otimes \left|i_2\right\rangle \otimes \cdots \otimes \left|i_n\right\rangle = \left|i_{\pi^{-1}1}\right\rangle \otimes \left|i_{\pi^{-1}2}\right\rangle \otimes \cdots \otimes \left|i_{\pi^{-1}1}\right\rangle . \tag{4.22}$$

A straightforward basis for the algebra of invariant operators can be obtained by applying the Reynold's operator in Eq. equation 4.2 to the Pauli basis. Normalizing such that all operators A_i are sums of unit norm Pauli terms, we obtain

$$\begin{aligned} A_{\boldsymbol{i}} &= \frac{1}{i_1! i_x! i_y! i_z!} \cdot \\ &\sum_{\pi \in \mathrm{S}_n} R(\pi) \left(\sigma_1^{\otimes i_1} \otimes \sigma_x^{\otimes i_x} \otimes \sigma_y^{\otimes i_y} \otimes \sigma_z^{\otimes i_z} \right) R^{-1}(\pi) \,, \end{aligned} \tag{4.23}$$

for every 4-tuple of positive integers

$$\boldsymbol{i} = (i_1, i_x, i_y, i_z) : i_1 + i_x + i_y + i_z = n \,, \tag{4.24}$$

where σ denote Pauli operators, and $\sigma_1 = \mathrm{id}_2$.

The dimension of the algebra X is of order $\mathrm{O}(n^3)$, and the previously stated theorems can be applied, reducing the naive ground state algorithm for permutation-invariant Hamil-

tonians on n qubits from an exponential to a polynomial runtime in n. This is formalized below.

Corollary 4.4.1. *The ground state energy of a permutation-symmetric Hamiltonian on n qubits, given as h_i in the basis of symmetrized Pauli monomials above, can be computed in time $\mathrm{O}\left(n^{3\omega}\right)$ via Theorem 4.3.1.*

Proof. All that is needed for applying Theorem 4.3.1 are the structure constants of the algebra X, which are computed in the Supplementary Information. □

One can similarly find the ground state of such an H efficiently as well. The output of the classical algorithm is a classical description of the state which can be efficiently constructed on a quantum computer via the Schur transform [31].

Corollary 4.4.2. *The ground state and ground state energy of a permutation-invariant Hamiltonian on n qubits, given as h_i in the basis of symmetrized Pauli monomials above, can be computed in time $\mathrm{O}\left(n^{\omega+1}\right)$ via Theorem 4.3.2.*

Proof. To apply Theorem 4.3.2, we must know the action of $A(X)$ on nontrivial eigenvectors of its projectors onto irreps. These eigenvectors are just the Schur basis [32]; we discuss this basis in more detail in the Supplementary Information, where we also explicitly give analytical expressions for matrix elements of $A(X)$. It is then easy to see that $\dim_X(\lambda) = \mathrm{O}(n)$, and also that the number of irreps with nonzero multiplicity is $\mathrm{O}(n)$. From Theorem 4.3.2, we immediately see that this gives an $\mathrm{O}\left(n^{\omega+1}\right)$-time algorithm for computing the ground state of S_n-equivariant Hamiltonians in the Schur basis. □

Remark 4.4.3. Though the structure constants $X_{\boldsymbol{k}}^{\boldsymbol{i},\boldsymbol{j}}$ and matrix elements $F_{q_\lambda,q'_\lambda}^{\boldsymbol{i},\lambda}$ for the completely symmetrized Pauli representation are problem independent, it is important to note that runtimes for evaluating the analytical expressions can be expensive polynomials in n that may matter in practice. Namely, we give expressions for the structure constants that take a total time $\mathrm{O}\left(n^{15}\right)$ and matrix elements that take a total time $\mathrm{O}\left(n^{10}\right)$ to evaluate numerically. We leave more efficient evaluations of these to future work.

Finally, we consider an application of Theorem 4.3.3 to the symmetric group case. We note that the Schur transform on n qubits can be implemented up to an accuracy ϵ in time $\tilde{\mathrm{O}}\left(n \operatorname{poly}\log\left(\epsilon^{-1}\right)\right)$ [32], giving an efficient (approximate) implementation of V_{STO}. As a

specific application of this result, we now consider a learning problem for which a variational quantum algorithm was given in [292]. We emphasize that here, just as in Theorem 4.3.3, we do not require that the input states ρ_i respect the symmetries of the model.

Corollary 4.4.4 (Efficient classical simulation of permutation-invariant models)**.** *Consider a binary classification problem with labels $y_i \in \{-1, 1\}$ and empirical loss*

$$\hat{\mathcal{L}}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{i=1}^{M} y_i \ell_{\boldsymbol{\theta}}(\rho_i), \tag{4.25}$$

where $\ell_{\boldsymbol{\theta}}(\rho_i)$ is as in Eq. equation 4.16 with a $\boldsymbol{\theta}$-dependent U. $\hat{\mathcal{L}}$ can be estimated to additive error ϵ at P points in time

$$\tilde{\mathrm{O}}\left(M |O|_\infty^2 \epsilon^{-2} n^5 \log\left(\frac{P}{\delta}\right) + MPn^{\omega+1}\right) \tag{4.26}$$

with total probability of success at least $1 - \delta$.

Proof. This follows immediately from Theorem 4.3.3 with $\delta \to \frac{\delta}{P}$ by the union bound. □

Corollary 4.4.4 implies that the loss of these models can be estimated completely classically when the states ρ_i are given as certain classical shadows descriptions; in the Supplementary Information, we also show that this procedure is efficient when the ρ_i have efficient matrix product state descriptions, even if they do not respect the symmetries of the model. As a point of comparison, consider the runtime of using a variational quantum algorithm to perform this binary classification task. Assume the variational circuits are of depth $\Omega\left(n^3\right)$ as required in Theorem 3 of [292] to ensure convergence. Then—taking $\Omega\left(|O|_\infty^2 \epsilon^{-2}\right)$ samples for each measurement to achieve an overall shot noise of $\mathrm{O}(\epsilon)$—this yields an overall runtime of $\Omega\left(MP|O|_\infty^2 \epsilon^{-2} n^3\right)$. For P sufficiently large, compare this to the time $\mathrm{O}\left(MPn^{\omega+1}\right)$ algorithm found for the classical algorithm where, even if quantum states are given as input, a classical shadow representation can be measured in quantum depth only $\mathrm{O}\left(n \operatorname{poly}\log\left(\epsilon^{-1}\right)\right)$. Unlike the quantum algorithm, this algorithm can be parallelized over irreps (i.e. over n_λ) easily, giving an effective runtime $\mathrm{O}\left(MPn^\omega\right) = \mathrm{o}\left(MPn^3\right)$. Even for P small, given many QPUs capable of running depth $\sim n$ quantum circuits, the classical algorithm parallelizes more effectively than the quantum algorithm as the required shadow tomography can be parallelized over shots.

4.5 Conclusion

We have specified a general framework for classically simulating highly symmetric quantum systems. Specializing to the symmetric group, we showed that these techniques yield an efficient classical algorithm for finding the ground state of quantum systems obeying an S_n symmetry, evaluating dynamics, and simulating S_n-equivariant quantum machine learning models. We hope that this framework sets the foundations for the future study of classical characterizations of symmetric quantum systems.

Chapter 5

ProjUNN: Fast and Efficient Unitary Neural Networks

For a long time, concepts in physics have served as motivation for advancements in computer science [248]. Inspired along the same lines, we consider here the task of integrating unitary matrices into deep neural networks. Such unitary matrices are a clever way to help improve the stability of deep neural networks as they have all unit-norm eigenvalues. Techniques from quantum information and Lie theory will be prevalent throughout this work.

5.1 Introduction

Learning in neural networks can often be unstable when networks are very deep or inputs are long sequences of data [24, 339]. For example, vanilla recurrent neural networks (RNNs) have recurrent states that are evolved via repeated application of a linear transformation followed by a pointwise nonlinearity, which can become unstable when eigenvalues of the linear transformation are not of magnitude one. Unitary matrices, which have eigenvalues of magnitude one, can naturally avoid this issue and have been used as a means to overcome these so-called vanishing and exploding gradients [24, 186]. More recently, unitary convolutional layers have been similarly constructed to help build more stable deep networks that are norm-preserving in their transformations [226, 296].

In the RNN setting, prior algorithms to apply $n \times n$ unitary matrices in RNNs have parameterized matrices into layers of unitary or orthogonal transformations or parameterized the Lie algebra of the unitary or orthogonal group (see table 5.1). In the layer-wise setting,

Table 5.1: When training RNNs on inputs with sequence length T, projUNN achieves nearly optimal runtime complexity while maintaining full parameterization of the unitary manifold.

Model	Complexity of gradient step	Layers to fully parameterize[a]	Method of parameterization
EURNN (tunable, n layers) [186]	$O(Tn^2)$	$O(n)$	Sequence of rotations
oRNN (n layers) [249]	$O(Tn^2)$	$O(n)$	Sequence of householder reflections
full-capacity URNN [336]	$O(Tn^2 + n^3)$	1	Parameterized matrix entries [b]
expRNN [223]	$O(Tn^2 + n^3)$	1	Parameterized matrix in Lie algebra [b]
projUNN (our method)	$O(Tn^2 + kn^2)$	1	Parameterized matrix entries [c]

[a] layers needed to parameterize the full unitary space, [b] approximations exist which may reduce runtimes though these approximations are not implemented here and can significantly bias the gradient [223], [c] runtime shown for typical setting when $k \ll n$ where k is the rank of gradient updates

unitarity is enforced for all values of parameters, but many layers are required to form a composition that can recreate any desired unitary, *i.e.,* fully parameterizing an $n \times n$ unitary requires $O(n)$ layers. By parameterizing the Lie algebra [223, 181], algorithms perform better on common benchmarks but have the drawback that performing gradient optimization on an $n \times n$ unitary requires $O(n^3)$ operations generically per step. Though not an issue with the small to medium sized models used today, this $O(n^3)$ is still $O(n)$ slower than standard methods of forward- and back-propogation in RNNs.

Motivated by the feature that gradients in neural networks are typically approximately low rank, we show that gradient updates to unitary/orthogonal matrices can be efficiently performed in low rank settings. We propose a new model called projUNN where matrices are first updated via gradient based optimization and then projected back onto the closest unitary (projUNN-D) or transported in the direction of the gradient (projUNN-T). projUNN has near-optimal runtime complexity unlike other existing algorithms for unitary RNNs (table 5.1) and is especially effective even in the most extreme case where gradients are approximated by rank one matrices. In RNN learning tasks, projUNN matches or exceeds benchmarks of state-of-the-art unitary neural network algorithms.

Though we present our model first in the RNN setting, we show that there is a direct extension of projUNN to the case of orthogonal/unitary convolution which we explore further. Here, we perform unitary/orthogonal convolution in the Fourier domain as inspired by [315]. Our algorithm runs efficiently in the convolutional setting especially for filters of large size and many channels (see appendix D.5 for more details).

5.2 Related works

Maintaining stability in neural networks via orthogonal or unitary matrices has a rich history of study in machine learning, both from an applied and theoretical perspective. Here, we briefly mention the most related works and algorithms we use in comparison to our projUNN. For a more holistic review of prior work in unitary neural networks and other related topics, please see appendix D.1.

Unitary neural networks were first designed to address the issue of vanishing and exploding gradients in RNNs while learning information in very long sequences of data more efficiently than existing parameterizations such as the long-short term memory unit (LSTM) [170]. Early algorithms [24, 249] maintained unitarity by constructing a series of parameterized unitary transformations. Perhaps the most effective of these methods is the efficient unitary recurrent neural network (EUNN) [186] which parameterized unitary matrices by composing layers of Givens rotations, Fourier transforms, and other unitary transformations. The unitary RNN (uRNN) of [336] and the Cayley parameterization (scoRNN) of [165] parameterized the full unitary space and maintained unitarity by performing a Cayley transformation. Later, [223] introduced the exponential RNN (expRNN) which parameterized unitary matrices in the Lie algebra of the orthogonal/unitary group. Though the uRNN, scoRNN, and expRNN perform well on benchmarks, their algorithms require matrix inversion or SVD steps which are time-consuming in high dimensions.

For convolutional neural networks, [296] showed how to efficiently calculate the singular values of a linear convolution and proposed an algorithm for projecting convolutions onto an operator-norm ball which relied on a series of costly projection steps. [226] introduced a block convolutional orthogonal parameterization (BCOP) which was faster and more efficient than the methods in [296], but required extra parameters in its parameterization and only parameterized a subset of the space of orthogonal convolutions. Most recently, [304] implemented orthogonal convolutions by parameterizing the Lie algebra of the orthogonal group via their skew orthogonal convolution (SOC) algorithm which approximates orthogonal convolutions especially well for small filter sizes. Finally, [315] performs convolutions in the Fourier domain via application of the Cayley transform. Our orthogonal/unitary convolutional parameterization is inspired by their approach and improves their runtime for convolutions over many channels.

5.3 Notation and background

Vectors and matrices are denoted with bold lower-case and upper-case script, $\boldsymbol{v}$ and $\boldsymbol{V}$, respectively. Scalars are denoted by regular script e and tensors are denoted by bold text $\mathbf{T}$. The complex conjugate of a complex-valued input $\cdot$ is denoted by $\cdot^*$ (ignored when real-valued). The transpose of a matrix $\boldsymbol{M}$ is denoted by $\boldsymbol{M}^\intercal$ and the conjugate transpose of a matrix is denoted by $\boldsymbol{M}^\dagger$. We denote the Frobenius norm of a matrix by $\|\cdot\|_F$ and the spectral norm of a matrix by $\|\cdot\|_2$.

Here, we provide a brief overview of the unitary/orthogonal groups and refer readers to section 1.2 for a more detailed mathematical background. The set of $n \times n$ orthogonal $O(n)$ and unitary $U(n)$ matrices are both Lie groups defined as

$$O(n) = \left\{\boldsymbol{M} \in \mathbb{R}^{n\times n} | \boldsymbol{M}\boldsymbol{M}^\intercal = \boldsymbol{I}\right\}, \qquad U(n) = \left\{\boldsymbol{M} \in \mathbb{C}^{n\times n} | \boldsymbol{M}\boldsymbol{M}^\dagger = \boldsymbol{I}\right\}. \tag{5.1}$$

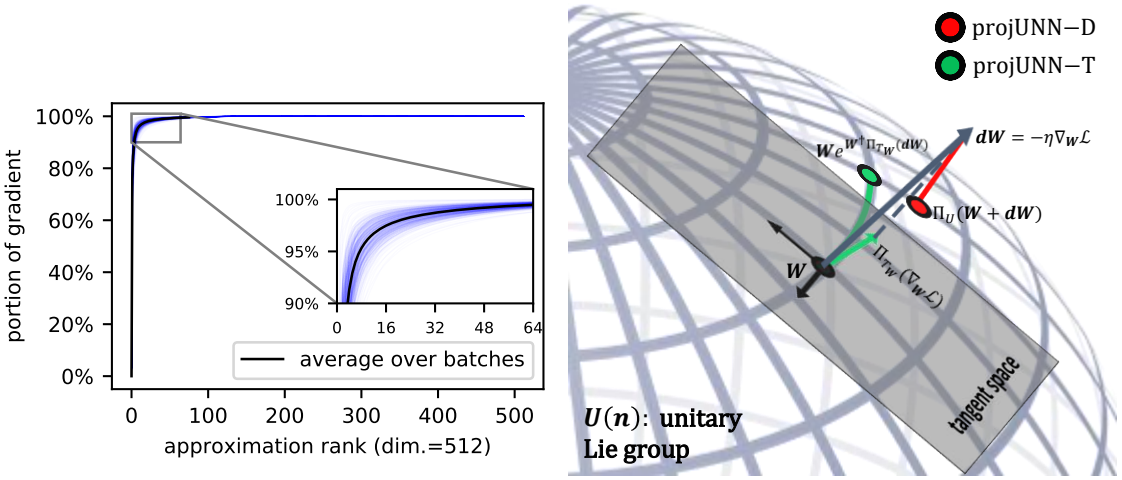
Constraining matrices in $O(n)$ and $U(n)$ to have determinant equal to one constructs the special orthogonal $SO(n)$ and unitary $SU(n)$ groups respectively. The Lie algebra or tangent space of the identity of $O(n)$ and $U(n)$ are the set of skew symmetric $\mathfrak{o}(n)$ and skew Hermitian $\mathfrak{u}(n)$ matrices,

$$\mathfrak{o}(n) = \left\{\boldsymbol{A} \in \mathbb{R}^{n\times n} : \boldsymbol{A} + \boldsymbol{A}^\intercal = 0\right\}, \qquad \mathfrak{u}(n) = \left\{\boldsymbol{A} \in \mathbb{C}^{n\times n} : \boldsymbol{A} + \boldsymbol{A}^\dagger = 0\right\}. \tag{5.2}$$

The matrix exponential $\exp(\cdot)$ is a map from the Lie algebra to the associated Lie group. The map is surjective if the Lie group is compact and connected – a property which holds for the unitary and special orthogonal groups but not the orthogonal group.

5.4 Projected unitary networks

Our projUNN algorithm is motivated by the simple observation that most of the "information" of a typical gradient in a deep learning task is captured in a low rank subspace of the complete gradient. fig. D-10 illustrates this feature when training our projUNN convolutional network on CIFAR10. We include further analysis and justification of this low rank behavior in appendix D.4. As we will show, we can perform updates on the low rank subspace of the gradient efficiently by approximating the gradient with a low rank matrix



(a) Low rank approximations

(b) Illustration of algorithm

Figure 5-1: (a) Low rank approximations capture most of the Frobenius norm of the gradient of a 512×512 matrix in the convolution filter (512 channels) of the last residual block of Resnet-9. Blue lines plot gradients of a single batch during training of our projUNN algorithm on CIFAR10 over a single epoch (see appendix D.4 for details and equivalent plot for RNN architecture). (b) Illustration of a single gradient update via gradient descent with learning rate η. projUNN-D (pictured in red) directly projects the gradient update back onto the unitary/orthogonal manifold. projUNN-T (pictured in green) first projects onto the tangent space (Lie algebra) and then performs a rotation in that direction via the exponential map.

and performing projections of parameters onto that low rank subspace. Our experiments show that this methodology, even with rank one approximations, is effective at learning and empirically introduces a form of "beneficial" stochasticity during gradient descent.

Based on how the projection is performed, our projUNN algorithm takes two forms illustrated in fig. 5-1b. The directly projected unitary neural network (projUNN-D) projects an update onto the closest unitary/orthogonal matrix in Frobenius norm. The tangent projected unitary neural network (projUNN-T) projects gradients onto the tangent space and transports parameters in that direction.

5.4.1 projUNN-D

projUNN-D takes advantage of the fact that the polar transformation returns the closest unitary or orthogonal matrix in the Frobenius norm to a given matrix (not necessarily unitary or orthogonal):

Lemma 5.4.1 (Projection onto unitary manifold [195]). *Given a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$:*

$$\Pi_U(\boldsymbol{A}) = \underset{\boldsymbol{U}\in\mathcal{U}(n)}{\arg\min} \|\boldsymbol{A} - \boldsymbol{U}\|_F^2 = \boldsymbol{A}(\boldsymbol{A}^\dagger \boldsymbol{A})^{-\frac{1}{2}}, \tag{5.3}$$

where $\mathcal{U}(n)$ indicates the set of $n \times n$ unitary matrices.

Note, that if the matrix $\boldsymbol{A}$ is real, then the projection above will be onto an orthogonal matrix. Given Lemma 5.4.1, projUNN-D performs optimization in two steps, which are illustrated in fig. 5-1b. First, matrix entries are updated via a standard learning step as in gradient descent, constructing a new matrix that is generally no longer unitary. In the second step, projUNN-D returns the unitary or orthogonal matrix closest in the Frobenius norm to the inputted matrix using lemma 5.4.1. At first sight, the second step would require $O(n^3)$ time to perform, but we can take advantage of the fact that gradient updates are typically approximately low rank (see appendix D.4). Efficient low rank approximations can be obtained using sampling methods detailed in section 5.4.3. With this in mind, we show that rank k updates can be performed in $O(kn^2)$ time when $k \ll n$.

Theorem 5.4.2 (Low rank unitary projection). *Let $\boldsymbol{U}$ be an $n\times n$ orthogonal/unitary matrix perturbed by $\boldsymbol{G}_k$, a rank k matrix. Then the projection onto the closest orthogonal/unitary matrix defined below can be performed in $O(k(n^2 + nk + k^2))$ steps.*

$$\boldsymbol{U} + \boldsymbol{G}_k \to \underset{\boldsymbol{V}\in\mathcal{U}}{\arg\min} \|\boldsymbol{U} + \boldsymbol{G}_k - \boldsymbol{V}\|_F^2. \tag{5.4}$$

To achieve this runtime, we perform updates completely in an $O(k)$ subspace of the full vector space. The operation $(\boldsymbol{U} + \boldsymbol{G}_k)[(\boldsymbol{U} + \boldsymbol{G}_k)^\dagger(\boldsymbol{U} + \boldsymbol{G}_k)]^{-1/2}$ can be decomposed into a series of $O(k)$ matrix-vector operations and an eigendecomposition of a $2k \times 2k$ sub-matrix. The complete proof and details are deferred to appendix D.2. One limitation of the above is that the eigendecomposition and inversion of a low rank matrix can cause numerical instability after many update steps. We discuss this further in appendix D.6.3 where we also provide options to alleviate this instability. projUNN-T, which we discuss next, does not require matrix inversion and is thus empirically more stable.

5.4.2 projUNN-T

projUNN-T maintains unitarity of matrices by orthogonally projecting gradient updates onto the tangent space and then performing a rotation in the direction of the projection (*i.e.,* along the geodesic). As in projUNN-D, there is a closed form for the orthogonal projection:

Lemma 5.4.3 (Tangent space projection [336])**.** *Given the tangent space $T_{\boldsymbol{U}}U(n)$ of an orthogonal/unitary matrix $\boldsymbol{U}$, the orthogonal projection $\Pi_{T_{\boldsymbol{U}}}$ with respect to the canonical metric $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \mathrm{Re}\left(\mathrm{Tr}[\boldsymbol{X}^\dagger \boldsymbol{Y}]\right)$ is*

$$\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}) = \frac{1}{2}\left(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{X}^\dagger \boldsymbol{U}\right). \tag{5.5}$$

Similar to lemma 5.4.1, this projection also returns the closest matrix in Frobenius norm to $\boldsymbol{X}$ in the tangent space,

$$\min_{\boldsymbol{Y} \in T_{\boldsymbol{U}}U(n)} \|\boldsymbol{Y} - \boldsymbol{X}\|_F = \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}). \tag{5.6}$$

Similar to projUNN-D, projUNN-T performs learning in two steps. First, a gradient update $\boldsymbol{G}$ is projected onto the tangent space using Lemma 5.4.3. Then, the orthogonal/unitary matrix is transported or rotated in the direction of the projection by application of the exponential map via the update rule [223, 336],

$$\boldsymbol{U} \to \boldsymbol{U} \exp\left[-\eta \boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G})\right], \tag{5.7}$$

where η denotes the learning rate. This update rule is an example of Riemannian gradient descent where we use the exponential map to transport gradient updates along the unitary/orthogonal manifold [54]. Here, we transport the matrix $\boldsymbol{U}$ along the geodesic in the direction of $\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G})$. This can be related to the update of projUNN-D which is an example of a retraction or an approximation to the exponential map of projUNN-T (see appendix D.2.3).

The update rule above requires matrix exponentiation and multiplication, both costly steps which can be sped up when $\boldsymbol{G}$ is a low rank matrix. Namely, to perform a rank k gradient update, we obtain an equivalent runtime scaling of $O(kn^2)$ for the projUNN-D when $k \ll n$.

Theorem 5.4.4 (Low rank tangent transport)**.** *Let $\boldsymbol{U}$ be an $n \times n$ orthogonal/unitary matrix perturbed by $\boldsymbol{G}_k$, a rank k matrix. Then projecting $\boldsymbol{G}_k$ onto the tangent space and performing a rotation in that direction as defined in eq.* (5.7) *can be performed in $O(k(n^2 + nk + k^2))$ steps.*

As with the ProjUNN-D, we achieve this runtime by performing the update above completely in an $O(k)$ subspace of the full vector space. The update via the exponential map can similarly be decomposed into a series of $O(k)$ matrix-vector operations and an eigendecomposition of a $2k \times 2k$ sub-matrix. Proper manipulations of the eigenvalues of the sub-matrix implement updates via the exponential map. The complete proof and details are deferred to appendix D.2.

5.4.3 Sampling methods

Commonly, gradients can have large rank but have still have many small singular values (*e.g.,* see fig. 5-1a). Here, a matrix $\boldsymbol{A}$ is deemed approximately low rank (see more details in appendix D.4), and one can obtain a rank k approximation $\boldsymbol{A}_k$ of $\boldsymbol{A}$ by sampling from rows and columns of $\boldsymbol{A}$. We use two sampling algorithms. The **LSI sampling** algorithm [267] obtains a rank k approximation to an $n \times n$ matrix $\boldsymbol{A}$ in time $O(kn^2 \log n)$. The algorithm projects the matrix $\boldsymbol{A}$ onto a random orthogonal subspace and then applies SVD based methods to the projected matrix to obtain the low rank approximation to that matrix. This algorithm features low approximation errors even for small k and is used extensively in our implementation. The **column sampling** (linear time SVD) algorithm [118] samples from the columns of an $n \times n$ matrix $\boldsymbol{A}$ to obtain a rank k approximation in $O(c^2 n + c^3)$ time, where c is a hyperparameter indicating the number of columns sampled. Typically, c is chosen as a multiple of k so the runtime is $O(k^2 n + k^3)$. In implementing this algorithm, we calculate the right singular vectors via matrix multiplication of the left singular vectors so the total runtime is $O(kn^2 + k^2 n + k^3)$.

We note that the two procedures described above, though sufficient for our purposes, can be further optimized in their asymptotic runtime. For sake of completeness, we discuss two of these other sampling algorithms in appendix D.4.

5.4.4 Extension to unitary or orthogonal convolution

Unitary/orthogonal convolutions are linear convolution operations that also preserve the 2-norm (isometric). Restricting convolutions to be unitary/orthogonal typically results in a drop in performance on standard imaging tasks when used in isolation, but prior work has explored unitary/orthogonal convolutions to potentially improve algorithmic stability and robustness (see appendix D.1.1 for more background) [226, 315]. We describe here how projUNN can be used to implement unitary/orthogonal convolutions in potentially a more efficient manner.

Given input tensor $\mathbf{X} \in \mathbb{C}^{M \times N \times C}$ where C is the number of channels of an $M \times N$ input, linear convolution (or technically cross-correlation) with a filter $\mathbf{W} \in \mathbb{C}^{M \times N \times C \times C}$ is defined as

$$[\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{p,q,d} = \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{n=1}^{N} \mathbf{W}_{m,n,d,c} \mathbf{X}_{p+m,q+n,c}, \tag{5.8}$$

where the indexing above is assumed to be cyclic (taken modulus the corresponding dimension) [220, 146]. Orthogonal/unitary convolutions form a subset of filters that preserve norms, *i.e.,* filters $\mathbf{W}$ such that $\|\mathrm{conv}_{\mathbf{W}}(\mathbf{X})\| = \|\mathbf{X}\|$. Equivalently, $\mathrm{conv}_{\mathbf{W}}(\cdot)$ is orthogonal/unitary if the Jacobian of the transformation is also orthogonal/unitary. To maintain unitarity/orthogonality, we set the dimensions of the filter $\mathbf{W}$ above such that it returns an output $\mathbf{Y}$ of the same dimension as the input $\mathbf{X}$. One can also perform semi-orthogonal or semi-unitary convolution by appropriately zero-padding an input or truncating from dimensions in the output.

Standard convolutional filters are typically supported over a sparse set of local elements, but performing orthogonal/unitary convolution generally requires implementing convolutions with filters supported over all elements resulting in slower runtimes. One can locally parameterize convolutional filters in the Lie algebra of the orthogonal/unitary group; nevertheless the exponential map into the Lie group expands the support of the filter:

$$\exp[\mathrm{conv}_{\mathbf{L}}](\mathbf{X}) = \mathbf{X} + \mathbf{L} * \mathbf{X} + \frac{1}{2}\mathbf{L} *^2 \mathbf{X} + \frac{1}{6}\mathbf{L} *^3 \mathbf{X} + \cdots \tag{5.9}$$

Thus, enforcing unitarity in convolutions generally requires additional overhead over the traditional setting of locally supported filters, but by performing convolution in the Fourier domain, runtimes for full-width filters can be optimally improved to $O(N^2 C \log(N) + N^2 C^2)$

[244]:

$$[\text{FFT}\,\text{conv}_{\mathbf{W}}(\mathbf{X})]_{\widehat{r},\widehat{s},:} = \widehat{\mathbf{W}}^{*}_{\widehat{r},\widehat{s},:,:}\ [\text{FFT}\,\mathbf{X}]_{\widehat{r},\widehat{s},:}\ , \tag{5.10}$$

where $\widehat{\mathbf{W}}_{i,j,:,:}$ is the value of the $\widehat{r}$ and $\widehat{s}$ frequency of $\mathbf{W}$ across all channels in the Fourier domain and FFT is the 2-dimensional fast Fourier transformation.

Our method is inspired by that of [315] which transformed $\mathbf{W}$ into Fourier space and performed a Cayley transformation (approximation to the exponential map into the Lie group) over the matrices indexed by $\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}$ which requires $O(N^2C^2\log(N) + N^2C^3)$ operations. For our algorithm, we parameterize $\mathbf{W}$ in the Fourier domain and only manipulate $\widehat{\mathbf{W}}$ (see appendix D.1.1 for a depiction of our parameterization). By parameterizing $\widehat{\mathbf{W}}$ directly and performing rank k updates using our projUNN, this runtime can be improved to $O(N^2C\log(N) + kN^2C^2)$ which is optimal when $k \ll N$. Our procedure for performing unitary/orthogonal convolution on an input $\mathbf{X}$ with filter $\mathbf{W}$ essentially follows the steps in eq. (5.10): perform an FFT on $\mathbf{X}$, block-multiply this by $\widehat{\mathbf{W}}$, and perform an inverse FFT on the output to obtain the final result.

Limitations Unitary/orthogonal convolutions are implemented in a cyclic fashion (*i.e.,* indices are taken modulus the dimension) which is not the standard approach but has been used before to accelerate convolutional operations [244]. Additionally, we parameterize convolution filters to have support over all possible elements (full-width), which can be expensive in memory. One can restrict the convolution to local terms in the Lie algebra, but this would not improve runtime as our algorithm runs in the Fourier space. To target local terms in a convolution, we instead propose for future work to implement a regularizer which has a specified support and penalizes the norm of the filter outside that support. Finally, the space of orthogonal convolutions has multiple disconnected components, which can present challenges for gradient based learning [226]. However, we can avoid this drawback by implementing projUNN using fully supported filters in the space of unitary convolutions which is connected (proof deferred to appendix D.2.4).

Theorem 5.4.5 (Unitary convolutional manifold is connected)**.** *The space of unitary convolutions with filters of full support has a single connected component.*

Algorithm 2 PROJUNN update step

Require: unitary matrix $\boldsymbol{U} \in \mathbb{C}^{N\times N}$ or orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{N\times N}$
Require: gradient update $\Delta\boldsymbol{U} \in \mathbb{C}^{N\times N}$ or $\Delta\boldsymbol{U} \in \mathbb{R}^{N\times N}$
Require: hyperparameter k corresponding to rank of approximation

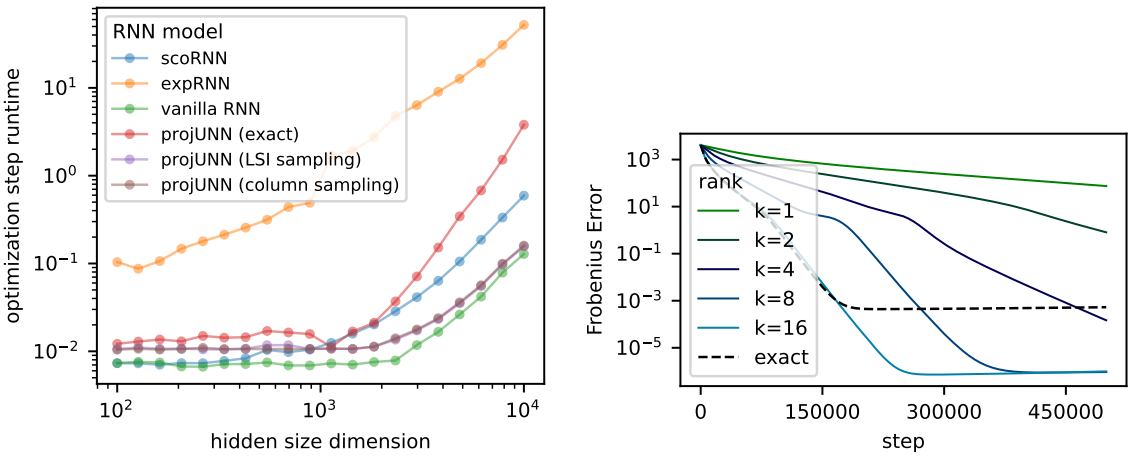
1: Obtain rank k approximation to $\Delta\boldsymbol{U}$ with output $\sum_{i=1}^{k} \boldsymbol{a}_i\boldsymbol{b}_i^\dagger \approx \Delta\boldsymbol{U}$ (see Section 5.4.3)
2: Follow steps in Theorem 5.4.2 (PROJUNN-D) or Theorem 5.4.4 (PROJUNN-T) in Appendix D.2:
3: Perform Gram-Schmidt (via QR decomposition) on concatenation of vectors $\boldsymbol{U}^\dagger\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ for all $i \in [k]$:
output $\boldsymbol{Q} \in \mathbb{C}^{N\times k}$ as semi-orthogonal matrix containing basis after Gram-Schmidt
4: Form matrix $\boldsymbol{K} \in \mathbb{C}^{2k\times 2k}$ below:
PROJUNN-D: $\boldsymbol{K} = \sum_{i=1}^{k} \boldsymbol{Q}^\dagger\boldsymbol{U}^\dagger\boldsymbol{a}_i\boldsymbol{b}_i^\dagger\boldsymbol{Q} + \boldsymbol{Q}^\dagger\boldsymbol{b}_i\boldsymbol{a}_i^\dagger\boldsymbol{U}\boldsymbol{Q} + \sum_{i=1}^{k}\sum_{j=1}^{k}(\boldsymbol{a}_i^\dagger\boldsymbol{a}_j)\boldsymbol{Q}^\dagger\boldsymbol{b}_i\boldsymbol{b}_j^\dagger\boldsymbol{Q}$
see Equation (D.23) *to Equation* (D.28)
PROJUNN-T: $\boldsymbol{K} = \frac{1}{2}\left[\sum_{i=1}^{k} \boldsymbol{Q}^\dagger\boldsymbol{U}^\dagger\boldsymbol{a}_i\boldsymbol{b}_i^\dagger\boldsymbol{Q} - \boldsymbol{Q}^\dagger\boldsymbol{b}_i\boldsymbol{a}_i^\dagger\boldsymbol{U}\boldsymbol{Q}\right]$
see Equation (D.34) *to Equation* (D.37)
5: Find eigenvalues $s_1, \ldots, s_{2k}$ and eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{2k}$ of $\boldsymbol{K}$
6: Perform update step by applying eigenvalue function:
PROJUNN-D: $\boldsymbol{U} \leftarrow (\boldsymbol{U} + \sum_{i=1}^{k} \boldsymbol{a}_i\boldsymbol{b}_i^\dagger)\left[\boldsymbol{I} + \sum_{j=1}^{2k}\left((s_j + 1 + \epsilon)^{-\frac{1}{2}} - 1\right)\boldsymbol{u}_j\boldsymbol{u}_j^\dagger\right]$
see Equation (D.30) *and Equation* (D.31), ϵ added for stability when $s_j \approx -1$ (we set $\epsilon = 10^{-8}$)
PROJUNN-T: $\boldsymbol{U} \leftarrow \boldsymbol{U}\left[\boldsymbol{I} + \sum_{j=1}^{2k}(\exp(-\eta s_j) - 1)\boldsymbol{u}_j\boldsymbol{u}_j^\dagger\right]$ where η is the learning rate
see Equation (D.39) *and Equation* (D.40)

5.4.5 Pseudocode for performing projUNN updates

Pseudocode for performing an update step on a unitary or orthogonal matrix $\boldsymbol{U}$ with a gradient update of $\Delta\boldsymbol{U}$ is shown in Algorithm 2. In convolutional settings, the steps in Algorithm 2 are applied across blocks of the convolution in Fourier space which can be performed in parallel. As a cautionary note, especially in the last step of Algorithm 2, where there is a composition of multiple matrix-vector multiplications, the order of these multiplications must be chosen to only perform matrix-vector operations to ensure optimal runtime. In other words, two $N \times N$ matrices should never be multiplied by each other at any point in this algorithm.

5.4.6 Runtime comparisons

PROJUNN has a nearly optimal asymptotic runtime scaling which offers practical benefits in high dimensions. In the RNN setting, fig. 5-2a shows that the low rank version of PROJUNN has a runtime that scales at the same rate as that of a vanilla RNN albeit with increased overhead. Updating the unitary matrix of PROJUNN takes $O(kn^2)$ time for performing updates of rank $k \ll n$, only a factor k more than a vanilla RNN which performs updates



(a) Runtime comparison

(b) Learning random unitary

Figure 5-2: (a) Runtime of PROJUNN (with low rank approximation) scales asymptotically at same rate of a vanilla RNN and much faster than other unitary RNN models or the exact version of PROJUNN (not using low rank approximation). Practical runtime improvements are achieved when the hidden dimension is large (see appendix D.5 for details). (b) PROJUNN-T can learn a random target unitary matrix using SGD. For a fixed learning rate, the loss decays at a rate proportional to the approximation rank k up to $k = 16$ where the approximation captures the full batch size (see exact PROJUNN which employs no approximation). The y-axis plots Frobenius error $\|\boldsymbol{U} - \boldsymbol{U}_{tar}\|_F^2$.

in $O(n^2)$ time. Note, that exact (full rank) updates to the $n \times n$ unitary matrices of a PROJUNN take roughly $O(n^3)$ time corresponding to the runtime of an SVD and equivalent to the runtime of expRNN and scoRNN [223, 165].

In the convolutional setting, PROJUNN offers the most benefit when there are many channels, filters with large support (very wide), or a need for exact unitary/orthogonal operations (in contrast with an approximate method like [304]). Given an $N \times N$ input with C channels, a forward and backward pass of PROJUNN runs in time $O(N^2C\log(N) + kN^2C^2)$ when performing rank k updates. This is a factor of C faster than the Cayley implementation [315] which runs in time $O(N^2C^2\log(N) + N^2C^3)$. For a more complete analysis of the asymptotic and empirical runtimes of various models including many not listed here, please see appendix D.5.

5.5 Experiments

We propose in this section a variety of benchmarked experiments to validate the efficiency and performance of the proposed projUNN method focusing mostly on RNN tasks.[1] We include further details of the experiments in Appendix D.3 including a preliminary empirical analysis of projUNN in convolutional tasks.

Toy model: learning random unitary To study the learning trajectories of projUNN, we consider a simple toy model aimed at learning a target random unitary. More specifically, we parameterize a large unitary matrix $\boldsymbol{U} \in \mathbb{C}^{2048\times 2048}$ to learn a Haar random target unitary $\boldsymbol{U}_{tar} \in \mathbb{C}^{2048\times 2048}$ given a dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i = \boldsymbol{U}_{tar}\boldsymbol{x}_i\}_{i=1}^{4096}$ of size 4096 where $\boldsymbol{x}_i \in \mathbb{C}^{2048}$ has entries drawn i.i.d. random normal. $\boldsymbol{U}$ is initialized as a random unitary matrix, and each step, we perform vanilla gradient descent over a batch of 16 training points using mean-squared error loss $\ell(\boldsymbol{x}_i, \boldsymbol{y}_i) = \|\boldsymbol{U}\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2$. Approximations of rank k to the gradient are obtained using the column sampling algorithm.

fig. 5-2b, which plots the Frobenius error $\|\boldsymbol{U}-\boldsymbol{U}_{tar}\|_F^2$, shows that projUNN-T equipped with the column sampling approximator is able to learn the random target unitary even when $k = 1$ (see appendix D.3.1 for plots with projUNN-D). Furthermore, for a fixed learning rate, learning requires fewer steps with larger k up to $k = 16$, the maximum rank of the gradient (note that $\nabla_{\boldsymbol{U}}\ell(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is rank 1). Therefore, approximating the gradient via low rank approximations can significantly speed up learning in this task (see appendix D.3.1 for further details).

Adding task In the adding task, an RNN must learn to add two numbers in a long sequence. We consider a variant of the adding task studied in [24], where the input consists of two data sequences of length T. The first is a list of T numbers sampled uniformly from $[0, 1]$, and the second is a list of binary digits set to zero except for two locations (those which must be summed) set to one located uniformly at random within the intervals $[1, T/2)$ and $[T/2, T)$ respectively.

Consistent with [165], we train our projUNN-T using an RNN with hidden dimension of 170 and the RMSprop optimizer to reduce the mean-squared error of the output with respect to the target. Naively predicting the average value of one for a random input achieves mean-

[1] code repository: https://github.com/facebookresearch/projUNN

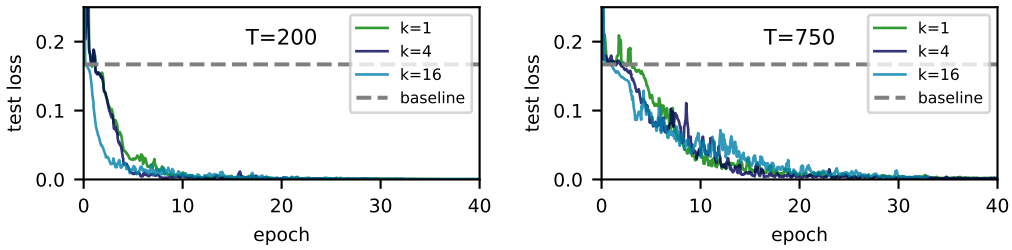


Figure 5-3: projUNN-T learns the adding task with $T = 200$ and $T = 750$. Test error is smoothed by taking the running average of 5 sequential points. See appendix D.3.2 for more details.

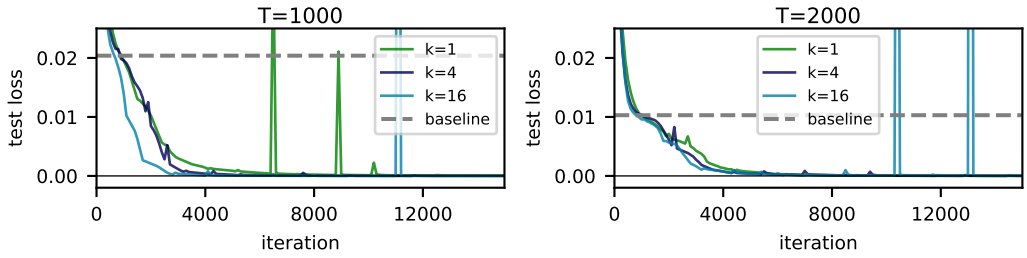


Figure 5-4: projUNN-T equipped with the column sampling approximation learns the copy task with $T = 1000$ and $T = 2000$ even with rank one approximations.

squared error of approximately 0.167. As shown in fig. 5-3, projUNN-T is able to learn the target function even with rank $k = 1$ approximations. Surprisingly, for a fixed learning rate and scheduler, convergence to the true solution is almost equally fast for $k = 1$, $k = 4$, and $k = 16$. Further details are provided in appendix D.3.2.

Copy memory task The copying memory task is a common benchmark for RNNs [170, 24, 166], where the aim is to memorize input data by ignoring a long sequence of void tokens. Given an alphabet of $n + 2$ symbols $\{a_i\}_{i=1}^{n+2}$, n of which represent data (sequence of letters $A, B, \dots$) and additional *void* (-) and *start recall* (:) tokens, the RNN must output the first K input tokens as the last K output tokens and *void* otherwise. An example input/output for $M = 6$ with $n = 4$ is

```
Input:  ABCDAD---·····-------:-----
Output: ----------·····-------ABCDAD
```

Here, $T = 1000$ or $T = 2000$ so the network must memorize data over a very long sequence of void tokens. As in [186], we consider $n = 8$ and input length $K = 10$ and train networks with

Table 5.2: Result of gradient descent optimization using the RMSprop optimizer on a single layer RNN for the permutedMNIST classification task. Each result is averaged over 3 runs, the same cross validation is done for all settings and includes the learning rate and its schedule. Training occurs for 200 epochs, and 10% of the training set (same for all models) is set apart as validation set. The training curves are provided in fig. D-8.

								projUNN-D					projUNN-T				
Width	RGD	LSTM	ScoRNN	ExpRNN	DT_∞	DT_{100}	DT_1	k=1	2	4	8	16	k=1	2	4	8	16
116	92.5	91.8	-	-	-	-	-	92.8	93.0	93.0	92.9	**93.2**	92.5	92.6	92.5	93.0	92.8
170	-	92.0	94.8	94.9	95.0	95.1	**95.2**	94.3	94.3	94.4	94.7	94.3	94.4	94.3	94.4	94.1	94.3
360	93.9	92.9	96.2	96.2	**96.5**	96.4	96.3	96.4	96.4	96.3	96.3	**96.5**	96.3	96.3	96.4	96.2	96.4
512	94.7	92.0	96.6	96.6	96.8	96.7	96.7	**97.0**	**97.0**	96.8	96.9	**97.0**	96.7	96.7	96.8	96.8	96.7

batch size 128 using the RMSProp algorithm. Naively predicting $T+K$ void tokens followed by K random selections of the n possible tokens achieves a baseline loss of $K\log(n)/(T+2K)$. projUNN-T is able to learn the copy task efficiently as shown in fig. 5-4. In fact, for fixed learning rates, rank one approximations using the column sampling algorithm provide the fastest convergence to the true solution in comparison to higher rank approximations. Networks were initialized using Henaff initialization (see appendix D.6.4) and the learning rate for unitary parameters was set to 32 times less than that of regular parameters (see appendix D.3.3 for more details).

Permuted MNIST Another challenging long-term memory task we consider is the permuted pixel-by-pixel MNIST dataset. Here, MNIST images are flattened, and pixels are randomly shuffled and placed in a sequence thereby creating some non-local dependencies. MNIST images have 28×28 resolution, so the pixel-by-pixel sequences have length $T = 784$. The task is digit classification (10 classes) as in standard MNIST models. We employ the same data processing, shuffle permutation, and formatting as that in prior works [223]. We perform cross-validation over different learning rates and evaluate both projUNN-T and projUNN-D with different low-rank values $k \in \{1, 2, 4, 8, 16\}$. The final test accuracy is shown in table 5.2. As observed in the copy and adding tasks, we find that using $k > 1$ does not lead to improved performances. In fact, we provide the evolution of the test set accuracy during training in fig. D-8 and note that as the number of updates is large (hundreds per epoch), even rank $k = 1$ update are able to move the model's parameters to their local optimum.

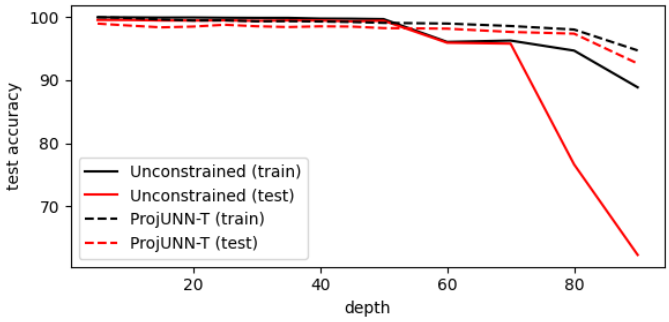


Figure 5-5: projUNN can more stably train very deep CNNs. Training on MNIST is done for 50 epochs in all cases with conv2d-BN-ReLU blocks (repeated "depth" times) and learning rate cross-validation (RMSprop), 32 channels throughout, and a final linear classifier. For 100 epochs and a depth of 100, we obtain $92.7, 23.5$ for the train/test accuracy of unconstrained CNN, and $95.7, 94.6$ for projUNN-T.

CNN experiments To explore the performance of our projUNN training algorithm for convolutional layers, we first analyzed its performance on CIFAR10 classification using a Resnet architecture [163]. Our aim was not to "beat" benchmarks but to provide an honest comparison of the performance of projUNN to existing methods. In fact, as noted earlier, enforcing unitarity generically results in a drop in accuracy for commonly used architectures. Consistent with prior work [315] we employ data-augmentation of random translations and left-right flips. Previous analysis in the RNN setting showed that rank $k = 1$ is sufficient for convergence so we always set $k = 1$ when using projUNN in the convolutional setting. For Resnet9 trained using the RMSprop optimizer, projUNN-T and projUNN-D reached 80.75% and 82.06% accuracy respectively, matching or outperforming reported results from existing unitary CNN models which achieved accuracies of 80.72% for BCOP [226] and 81.70% for Cayley [315] (further details in Appendix D.3.5). Note, that all of these methods resulted in a performance drop compared to the standard model (without unitary constraints) which achieved accuracy of 92.26%. Hence, we believe that there remain a large potential for unitary models to close this gap. Separate from just performance and to motivate the use of unitary parameterization, we provide in fig. 5-5, test accuracy results from a simple CNN model with progressively increasing depth trained with and without unitary parameterization on MNIST data. We observe that unitary weights might provide benefits for vanilla CNN architectures that have not been designed to handle very deep settings. Of course, various techniques and tricks have been designed to enable CNNs to be trainable at large depths [339, 163, 52]. Unitary convolutions, which are simple and theoretically motivated, can potentially be used either separately or in-tandem with these other techniques.

5.6 Discussion

Our PROJUNN shows that one need not sacrifice performance or runtime in training unitary neural network architectures. Our results broadly take advantage of the approximate low rank structure of parameter gradients to perform updates at nearly optimal runtime. Looking beyond the setting studied here, it is an interesting question how our framework can be applied to other neural network architectures or parameter manifolds. Group convolutional neural networks and Riemannian gradient descent offer two promising avenues for further application of our techniques.

Appendix A

Appendix for Chapter 2

A.1 Training Error Dominates in the Optimization of Variational Quantum Algorithms

The variational quantum eigensolver [273] is purely a problem of optimization and may appear unrelated to the challenges in learning via variational algorithms; however, by decomposing the error of a learning algorithm into key terms using well-established methods [55], we will show that variational learning algorithms essentially face the same optimization task and its associated challenges. In both cases, the hardness of learning or optimizing with variational circuits manifests itself in the challenges of optimizing over a cost landscape riddled with traps (or other barriers to optimization).

We restrict ourselves here to the supervised learning framework of empirical risk minimization, where our goal is to learn a space of input and output pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ drawn from a distribution $P(\boldsymbol{x}, \boldsymbol{y})$. Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, we quantify how well our function performs by considering the expected risk $\mathcal{R}$:

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}} \left[\, \ell(f(\boldsymbol{x}), f^*(\boldsymbol{x})) \, \right], \tag{A.1}$$

where the expectation above is taken with respect to $P(\boldsymbol{x}, \boldsymbol{y})$. To benchmark performance, we compare to the "optimal" or target function f^* which is the minimizer of the risk:

$$f^*(\boldsymbol{x}) = \arg\min_{\hat{\boldsymbol{y}} \in \mathcal{Y}} \mathbb{E} \left[\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) | \boldsymbol{x} \right]. \tag{A.2}$$

To perform learning, we search for a function $\widehat{f} \in \mathcal{F}$ in the function class $\mathcal{F}$ (think *e.g.,* the set of functions expressed by quantum neural networks). The expected risk $\mathcal{R}(f)$ is not something one can calculate since it requires access to the full probability distribution of the data. Instead, one minimizes the empirical risk $\widehat{\mathcal{R}}(f)$ (often named the training error) over a given training data set $\mathcal{D}$ of size N consisting of pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$:

$$\widehat{\mathcal{R}}(f) = \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), f^*(\boldsymbol{x}_i)). \tag{A.3}$$

Note that we use the hat in $\widehat{\mathcal{R}}$ and $\widehat{f}$ to denote the expected risk measure and function that one actually has access to during training or optimization. Given the above, one can bound the expected risk of any function $\widehat{f}$ as a decomposition below [55]:

$$\mathbb{E}\left[\mathcal{R}(\widehat{f}) - \mathcal{R}(f^*)\right] \leq \underbrace{\min_{f\in\mathcal{F}} \mathcal{R}(f) - \mathcal{R}(f^*)}_{\text{approximation error}} + \underbrace{2\mathbb{E}\left[\sup_{f\in\mathcal{F}} \left|\mathcal{R}(f) - \widehat{\mathcal{R}}(f)\right|\right]}_{\text{generalization error}} + \underbrace{\mathbb{E}\left[\widehat{\mathcal{R}}(\widehat{f}) - \min_{f\in\mathcal{F}} \widehat{\mathcal{R}}(f)\right]}_{\text{optimization error}}, \tag{A.4}$$

where the expectation above is taken with respect to the distribution over data sets or training sets. The proof of this statement follows by a careful, yet straightforward, application of additions/subtractions with corresponding bounds [55].

Proof. Let $\widehat{f}_{\mathcal{F}} = \arg\min_{f\in\mathcal{F}} \widehat{\mathcal{R}}(f)$ and $f_{\mathcal{F}} = \arg\min_{f\in\mathcal{F}} \mathcal{R}(f)$. Then, by adding and subtracting quantities, we obtain the following result:

$$\begin{aligned}\mathbb{E}\left[\mathcal{R}(\widehat{f}) - \mathcal{R}(f^*)\right] = \mathbb{E}\big[&\mathcal{R}(\widehat{f}) - \mathcal{R}(f^*) \\ &+ \widehat{\mathcal{R}}(\widehat{f}_{\mathcal{F}}) - \widehat{\mathcal{R}}(\widehat{f}_{\mathcal{F}}) \\ &+ \mathcal{R}(f_{\mathcal{F}}) - \mathcal{R}(f_{\mathcal{F}}) + \widehat{\mathcal{R}}(f_{\mathcal{F}}) - \widehat{\mathcal{R}}(f_{\mathcal{F}}) \\ &+ \widehat{\mathcal{R}}(\widehat{f}) - \widehat{\mathcal{R}}(\widehat{f})\big].\end{aligned} \tag{A.5}$$

We reorder the above as follows and note their relation to the main statement:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{R}(\widehat{f}) - \mathcal{R}(f^*)\right] = \; & \mathbb{E}\left[\mathcal{R}(f_{\mathcal{F}}) - \mathcal{R}(f^*)\right] && \text{approximation error} && \text{(A.6)}\\
& + \mathbb{E}\left[\mathcal{R}(\widehat{f}) - \widehat{\mathcal{R}}(\widehat{f})\right] && \text{generalization error} && \text{(A.7)}\\
& + \mathbb{E}\left[\widehat{\mathcal{R}}(f_{\mathcal{F}}) - \mathcal{R}(f_{\mathcal{F}})\right] && \text{generalization error} && \text{(A.8)}\\
& + \mathbb{E}\left[\widehat{\mathcal{R}}(\widehat{f}_{\mathcal{F}}) - \widehat{\mathcal{R}}(f_{\mathcal{F}})\right] && \leq 0 \text{ since } \widehat{f}_{\mathcal{F}} \text{ minimizes } \widehat{\mathcal{R}} && \text{(A.9)}\\
& + \mathbb{E}\left[\widehat{\mathcal{R}}(\widehat{f}) - \widehat{\mathcal{R}}(\widehat{f}_{\mathcal{F}})\right] && \text{optimization error} && \text{(A.10)}
\end{aligned}$$

For the quantities in the generalization error, we have since $\widehat{f}, f_{\mathcal{F}} \in \mathcal{F}$:

$$\begin{aligned}
\mathbb{E}\left[\mathcal{R}(\widehat{f}) - \widehat{\mathcal{R}}(\widehat{f})\right] &\leq \mathbb{E}\left[\sup_{f \in \mathcal{F}} \left|\mathcal{R}(f) - \widehat{\mathcal{R}}(f)\right|\right] \\
\mathbb{E}\left[\widehat{\mathcal{R}}(f_{\mathcal{F}}) - \mathcal{R}(f_{\mathcal{F}})\right] &\leq \mathbb{E}\left[\sup_{f \in \mathcal{F}} \left|\mathcal{R}(f) - \widehat{\mathcal{R}}(f)\right|\right].
\end{aligned} \tag{A.11}$$

Plugging these into equation A.5 and noting as before that $\mathbb{E}\left[\widehat{\mathcal{R}}(\widehat{f}_{\mathcal{F}}) - \widehat{\mathcal{R}}(f_{\mathcal{F}})\right] \leq 0$, we arrive at the desired result.

□

In the context of quantum variational algorithms, each of these has the following properties:

- The **approximation error** quantifies how well the most optimal function in the hypothesis class $\mathcal{F}$ can fit the function. In variational settings, the approximation error is typically bounded by assuming the target function is generated from a nice class of functions (e.g. shallow circuits) or arguing either analytically or theoretically that a given ansatz can (approximately) express the target function [255, 303, 120, 300].

- The **generalization error** quantifies the statistical error that arises from having a finite data set and is typically insignificant in quantum variational algorithms where circuit complexity is limited with regards to the number of training samples. More precisely, for data sets of size m, previous work [69, 121] bound the generalization error as $\tilde{O}(\sqrt{|G|/m})$ where $|G|$ is the number of trainable gates. In contrast, generalization error in heavily overparameterized classical neural network models are challenging

to bound and it is still an open question why deep learning models generalize so well [351, 258].

- The **optimization error** measures how well one is able to reduce the empirical risk. Issues with optimization such as poor local minima and barren plateaus arise here. Note that there is a distinct difference between quantum and classical deep learning here. With classical deep neural networks, this quantity is typically negligible since neural networks are overparameterized with respect to the data set size and can fit random data arbitrarily well [351, 240]. Furthermore, due to efficient means of calculating gradients with bit-level precision, classical machine learning algorithms perform optimization over parameters far more efficiently than quantum variational algorithms. In quantum variational models, overparameterization with respect to the Hilbert space dimension is generally needed to arbitrarily fit data [20, 335, 206]. Since the Hilbert space dimension grows exponentially with the number of qubits, such overparameterization becomes prohibitive rather rapidly.

In summary, the approximation error and generalization error can be bounded efficiently with sufficient data so failures in learning are typically related to optimization over the empirical risk. As an aside, this is loosely analogous to the classical setting of learning polynomial size Boolean circuits which is strongly conjectured to be hard since the space of Boolean functions is challenging to search over [22].

Finally, we would like to stress that the decomposition of the excess risk performed in this section is neither unique nor necessarily tight. The decomposition can be performed in various other ways depending on the quantities one would like to bound. We chose the decomposition here to relate errors in quantum machine learning algorithms to their classical counterparts and to highlight the challenges one may face when attempting to provably learn a target function class.

A.2 Statistical Query Framework: Background and Additional Details

The statistical query (SQ) framework was introduced nearly 25 years ago to analyze the hardness of learning problems [194]. This framework restricts algorithms to a series of noisy

queries, and hardness results are stated in terms of the number of queries needed to learn a given class of functions. Since there are various different ways of defining the statistical query model—including a recent quantum oracular version proposed in [30]—let us first review some of the various models considered in prior work.

1. **Classical statistical query model**: Introduced by [194], this was the first statistical query model introduced. For a given distribution D of inputs over an input space X and target concept $c : X \to \{-1, +1\}$, one can make a statistical query $\mathrm{SQ}(q, \tau)$, by providing a threshold $\tau \in \mathbb{R}^+$ and a query function $q : X \times \{-1, +1\} \to \{-1, +1\}$. The query returns a value in the range:

$$\mathbb{E}_{x\sim D}\left[q(x, c(x)) - \tau\right] \leq \mathrm{SQ}(q, \tau) \leq \mathbb{E}_{x\sim D}\left[q(x, c(x)) + \tau\right]. \tag{A.12}$$

2. **Correlational statistical query model**: The query is the same as before, except now, one queries correlations $\mathrm{CSQ}(q, \tau)$ only by providing a threshold $\tau \in \mathbb{R}^+$ and a query function $h : X \to \{-1, +1\}$. The query returns a value in the range:

$$\mathbb{E}_{x\sim D}\left[h(x)c(x) - \tau\right] \leq \mathrm{CSQ}(h, \tau) \leq \mathbb{E}_{x\sim D}\left[h(x)c(x) + \tau\right]. \tag{A.13}$$

   This model is strictly less powerful than the standard statistical query model since one can perform a correlational statistical query with a standard statistical query [312].

3. **Quantum statistical query model**: This is a statistical query model with quantum samples [30]. Here, we are restricted to target (classical) Boolean functions $c : \{0, 1\}^n \to \{-1, +1\}$. A quantum statistical query $\mathrm{Qstat}(\tau, M)$ is provided with a threshold $\tau \in \mathbb{R}^+$ and an observable or Hamiltonian $M \in (\mathbb{C}^2)^{n+1} \times (\mathbb{C}^2)^{n+1}$ satisfying $\|M\| \leq 1$ and returns a number in the range:

$$\langle\psi_c| M |\psi_c\rangle - \tau \leq \mathrm{Qstat}(M, \tau) \leq \langle\psi_c| M |\psi_c\rangle + \tau, \tag{A.14}$$

   where $|\psi_c\rangle = \sum_{x\in\{0,1\}^n} \sqrt{D(x)}\,|x\rangle\,|c(x)\rangle$. This model is useful to analyze the hardness of learning classical Boolean functions when given the extra power of querying the classical function in superposition. Our work considers learning quantum data and thus does not fit into the framework of this SQ model.

The SQ learning setting is related to the probably approximately correct (PAC) setting of learning theory [316] in that if an algorithm can learn a given function class in the SQ learning setting under any input distribution, then that function class is also PAC learnable [194, 283]. Two very recent works have studied the SQ hardness of learning data generated by quantum circuits. First, [169] analyze the hardness of learning the output distribution of clifford circuits and stabilizer states showing that these distributions are hard to learn using classical Boolean SQ oracles. Nevertheless, when given samples from the Boolean hypercube of the distribution, they provide an efficient algorithm based on linear regression to determine the stabilizer state underlying the distribution. Such a result is similar to classic results in [194] showing that parity functions are hard to learn using only SQ oracle calls but easy when performing linear regression with enough samples. Second, [141] show that learning stabilizer states is hard in an SQ setting where queries are made over two-outcome POVMs. Their results show that learning stabilizer states in such a setting is as hard as learning the function class of parity with noise in the standard Boolean setting. Our results expand the set of quantum functions that are hard to learn in SQ settings and relate such hardness results to the variational setting.

A.2.1 Quantum Statistical Query Models

Variational quantum algorithms are inherently noisy due to unavoidable sources such as the need for sampling outputs, or potentially correctable sources such as gate errors and state preparation noise. In such noisy settings, the statistical query (SQ) model provides a useful framework for quantifying the complexity of learning a class of functions by considering how many query calls to a noisy oracle are needed to learn any function in that class. As described in the main text, in the variational setting, we consider two forms of statistical queries which relate to learning a target Hamiltonian or a target unitary, both of which result in exponential hardness results for learning simple variational classes of data.

Our proofs expand on recent research showing hardness results in the SQ setting for certain quantum machine learning problems. More specifically, recent results that have shown that certain fundamental and rather simple classes of quantum "functions" are hard to learn in the SQ setting. Namely, (classical) output distributions of locally constructed quantum states [169] and the set of Clifford circuits [141] are hard to learn given properly chosen statistical query oracles. Following these results, we show that simple classes of

functions generated by variational circuits are also exponentially difficult to learn in the SQ settings we consider. We also directly connect the statistical query setting to actual optimization algorithms that are used in practice for variational optimization. Our results indicate that training algorithms must be carefully constructed to avoid these poor lower bounds.

A.2.2 Limitations of Hardness Results in the SQ Framework

Though the SQ framework is a useful tool for analyzing the hardness of learning a class of functions in noisy settings, there are a few caveats and limitations of any hardness results proven in the SQ setting:

- The statistical query model inherently requires noise in the form of the tolerance τ. Furthermore, the guarantees of learning must handle worst case noise scenarios where the noise acts adversarially on the statistical query. Though quantum variational algorithms are inherently noisy, this noise typically does not arise in an adversarial nature.

- The statistical query model places bounds on learning classes of functions using optimizers that query this SQ model and is not directly related to issues of loss landscapes since there is no loss landscape to actually optimize. Nevertheless, since (noisy) calculations of gradients and loss function values are themselves examples of statistical queries, any issues with optimizing over a loss landscape will also arise in performing the optimizer through a series of statistical queries.

- Learning every function in a class $\mathcal{C}$ can be restrictive, and in practice, one may only really want to learn a given function or a small set of functions. In fact, it can be shown that even the class of functions generated by shallow neural networks is hard to learn in the SQ setting [115, 139, 140, 87, 114]; nevertheless, neural networks are very successful at learning specific functions such as the classification of real-world images [221].

- Specific to the settings considered here, our hardness results were obtained in the correlational SQ setting by constructing a family of orthogonal functions drawn from a given function class. We chose this setting for its close relation to the algorithms

used in practice for performing optimization over variational parameters. However, as mentioned in the main text, the correlational SQ setting is strictly weaker than the more general SQ setting, and separations between SQ and correlational SQ results have been made in prior work [85, 13].

A.3 Proofs of Statistical Query Results

Throughout this section, we make use of standard formulas from Weingarten calculus to integrate over Haar measure or t-designs [256, 95, 96]. Let $|I_m^n\rangle$ denote n copies of the unnormalized maximally entangled state on a Hilbert space of dimension m:

$$|I_m^n\rangle = \sum_{i_1,i_2,\ldots,i_n=1}^{m} |i_1, i_2, \ldots, i_n\rangle\, |i_1, i_2, \ldots, i_n\rangle\,. \tag{A.15}$$

For $n = 2$, let $|S_m^2\rangle$ denote the same unnormalized state as above with a swap operation applied to the second register:

$$\begin{aligned} |S_m^2\rangle &= (\boldsymbol{I} \otimes \mathrm{SWAP})\, |I_m^2\rangle \\ &= \sum_{i_1,i_2=1}^{m} |i_1, i_2\rangle\, |i_2, i_1\rangle\,. \end{aligned} \tag{A.16}$$

The following hold over a distribution $\mathcal{D}$ that is a 2-design over the unitary matrices of dimension m:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\boldsymbol{U}\otimes\bar{\boldsymbol{U}}\right] &= \frac{1}{m}\,|I_m^1\rangle\langle I_m^1|\,, \\ \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\boldsymbol{U}\otimes\boldsymbol{U}\otimes\bar{\boldsymbol{U}}\otimes\bar{\boldsymbol{U}}\right] &= \frac{1}{m^2-1}\left(|I_m^2\rangle\langle I_m^2| + |S_m^2\rangle\langle S_m^2|\right) - \frac{1}{m(m^2-1)}\left(|I_m^2\rangle\langle S_m^2| + |S_m^2\rangle\langle I_m^2|\right), \end{aligned} \tag{A.17}$$

where $\bar{\boldsymbol{U}}$ denotes the matrix with entries that are the complex conjugate of entries of $\boldsymbol{U}$.

As a simple example of applying the techniques above, we show that for unitaries $\boldsymbol{U}_*$ and $\boldsymbol{V}$ of dimension d^n (e.g., $d = 2$ for qubits and n is the number of qubits), $\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger\boldsymbol{V}\rho)]\right] = d^{-n}\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger\boldsymbol{V})]$ whenever $\mathcal{D}$ forms a 1-design. This is a crucial formula that we use in the evaluation of statistical queries to qUSQ.

Lemma A.3.1. *For any distribution $\mathcal{D}$ that is a 1-design over states of dimension d^n,*

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{W}^\dagger\boldsymbol{V}\rho)]\right] = d^{-n}\Re[\mathrm{Tr}(\boldsymbol{W}^\dagger\boldsymbol{V})]. \tag{A.18}$$

Proof. WLOG, we rewrite the equation above in terms of a distribution over pure states and with a slight abuse of notation, we let $\mathcal{D}$ also denote a distribution over unitary matrices $\boldsymbol{U}$ that forms a 1-design:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{W}^\dagger\boldsymbol{V}\rho)]\right] = \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\Re[\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{W}^\dagger\boldsymbol{V}\boldsymbol{U}\,|0\rangle]\right]. \tag{A.19}$$

Using equation A.15, we have:

$$\mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\Re[\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{W}^\dagger\boldsymbol{V}\boldsymbol{U}\,|0\rangle]\right] = \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\langle I^1_{d^n}|\left((\boldsymbol{W}^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\right)\left(\boldsymbol{U}\otimes\bar{\boldsymbol{U}}\right)|0\rangle\,|0\rangle\right]. \tag{A.20}$$

Applying equation A.17, we have:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\Re[\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{W}^\dagger\boldsymbol{V}\boldsymbol{U}\,|0\rangle]\right] &= \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\Re[\langle I^1_{d^n}|\left((\boldsymbol{W}^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\right)\left(\boldsymbol{U}\otimes\bar{\boldsymbol{U}}\right)|0\rangle\,|0\rangle]\right]\\ &= \frac{1}{d^n}\Re[\langle I^1_{d^n}|\left((\boldsymbol{W}^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\right)|I^1_{d^n}\rangle\langle I^1_{d^n}|\,|0\rangle\,|0\rangle] \\ &= \frac{1}{d^n}\Re[\mathrm{Tr}(\boldsymbol{W}^\dagger\boldsymbol{V})].\end{aligned} \tag{A.21}$$

□

A.3.1 Lower Bounds for Statistical Query Learning

For completeness, we include a proof of Theorem A.3.3, copied below, which lower bounds the number of queries needed to learn a hypothesis class. qCSQ and qUSQ both take the form of an inner product so the proof holds for both cases. As a reminder we include the definitions of qCSQ and qUSQ below.

Quantum correlational statistical query (qCSQ) Given a target observable $\boldsymbol{M}$ and a distribution $\mathcal{D}$ of input states, a query qCSQ$(\boldsymbol{O},\tau)$ takes in a bounded observable $\boldsymbol{O}$ with $\|\boldsymbol{O}\|\leq 1$ and a tolerance τ and returns a value in the range:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho)-\tau\right] \leq \mathrm{qCSQ}(\boldsymbol{O},\tau) \leq \mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{O}\rho)\,\mathrm{Tr}(\boldsymbol{M}\rho)+\tau\right]. \tag{A.22}$$

Quantum unitary statistical query (qUSQ) Given a target unitary transformation $\boldsymbol{U}_*$ over a distribution $\mathcal{D}$ of inputs, the oracle qUSQ$(\boldsymbol{V}, \tau)$ takes in a unitary matrix $\boldsymbol{V}$ and a tolerance τ and returns a value in the range:

$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V}\rho)] - \tau\right] \leq \mathrm{qUSQ}(\boldsymbol{V}, \tau) \leq \mathbb{E}_{\rho\sim\mathcal{D}}\left[\Re[\mathrm{Tr}(\boldsymbol{U}_*^\dagger \boldsymbol{V}\rho)] + \tau\right]. \tag{A.23}$$

Finally, we remind the reader of the definition of the statistical query dimension.

Definition A.3.2 (Statistical query dimension [53, 283])**.** For a distribution $\mathcal{D}$ and concept class $\mathcal{H}$ where $\|\boldsymbol{M}\|_{\mathcal{D}}^2 \leq C_{max}$ for all $\boldsymbol{M} \in \mathcal{H}$, the statistical query dimension (SQ-DIM$_{\mathcal{D}}(\mathcal{H})$) is the largest positive integer d such that there exists d observables $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_d \in \mathcal{H}$ such that for all $i \neq j : |\langle \boldsymbol{M}_i, \boldsymbol{M}_j\rangle_{\mathcal{D}}| \leq C_{max}/d$.

From here, we provide a proof of the query complexity of learning a function class, similar to the proof in [312].

Theorem A.3.3 (Query complexity of learning [312, 53])**.** *Given a distribution $\mathcal{D}$ on inputs and a hypothesis class $\mathcal{H}$ where $\|\boldsymbol{M}\|_{\mathcal{D}}^2 \leq C_{max}$ for all $\boldsymbol{M} \in \mathcal{H}$, let $d = $ SQ-DIM$_{\mathcal{D}}(\mathcal{H})$ be the statistical query dimension of $\mathcal{H}$. Any* qCSQ *or* qUSQ *learner making queries with tolerance $C_{max}\tau$ must make at least $(d\tau^2 - 1)/2$ queries to learn $\mathcal{H}$ up to error $C_{max}\tau$.*

Proof. Since we are restricted to the weaker setting of correlational statistical queries in this study, we can reuse a simple and elegant proof from [312].

Let $\boldsymbol{M}_1, \boldsymbol{M}_2, \cdots, \boldsymbol{M}_d$ be d functions that saturate SQ-DIM$_{\mathcal{D}}(\mathcal{H})$, i.e., $\langle \boldsymbol{M}_i, \boldsymbol{M}_j\rangle_{\mathcal{D}} \leq 1/d$ for all $i \neq j$. Assume we apply query $\boldsymbol{O}$ and let $S = \{i \in [d] : \langle \boldsymbol{O}, \boldsymbol{M}_i\rangle_{\mathcal{D}} > C_{max}\tau\}$ Then, by simple application of Cauchy-Schwarz, we have that for any query $\boldsymbol{O}$:

$$\begin{aligned}\left\langle \boldsymbol{O}, \sum_{i\in S} \boldsymbol{M}_i \right\rangle_{\mathcal{D}}^2 &\leq C_{max} \left\| \sum_{i\in S} \boldsymbol{M}_i \right\|_{\mathcal{D}}^2 \\ &= C_{max} \sum_{i,j\in S} \langle \boldsymbol{M}_i, \boldsymbol{M}_j\rangle_{\mathcal{D}} \\ &\leq C_{max}^2 \left(|S| + |S|^2/d\right).\end{aligned} \tag{A.24}$$

Note, that we can also bound the quantity above from below by using the definition of S:

$$\left\langle \boldsymbol{O}, \sum_{i\in S} \boldsymbol{M}_i \right\rangle_{\mathcal{D}} \geq C_{max}|S|\tau. \tag{A.25}$$

Combining the above, we have that

$$|S| \leq d/(d\tau^2 - 1). \tag{A.26}$$

Similarly, defining $S' = \{i \in [d] : \langle \boldsymbol{O}, \boldsymbol{M}_i \rangle_{\mathcal{D}} < -C_{max}\tau\}$ with correlation less than $-\tau$, we follow the steps above to also note that $|S'| \leq d/(d\tau^2 - 1)$. Altogether, we have that $|S'| + |S| \leq 2d/(d\tau^2 - 1)$, which implies that each oracle call returning 0 is inconsistent with at most $2d/(d\tau^2 - 1)$ functions. This results in the lower bound stated, as d functions must be ruled inconsistent to learn the target class. □

A.3.2 Proofs of Statistical Query Dimensions for Variational Function Classes

Proposition A.3.4 (SQ dimension for $L = 1$ and fixed global measurement)**.** *Given n qubits, let $\mathcal{H}$ be the concept class containing functions $f : \mathbb{C}^{2^n} \to \mathbb{R}$ consisting of single qubit rotations followed by a global Pauli Z measurement, i.e. functions of the form*

$$f(|\psi\rangle; \boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n) = \langle\psi| \left(\boldsymbol{U}_1^\dagger \otimes \boldsymbol{U}_2^\dagger \otimes \cdots \otimes \boldsymbol{U}_n^\dagger\right) (\boldsymbol{Z}_1 \otimes \boldsymbol{Z}_2 \otimes \cdots \boldsymbol{Z}_n) (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2 \otimes \cdots \otimes \boldsymbol{U}_n) |\psi\rangle, \tag{A.27}$$

where $|\psi\rangle$ is the input to the function and $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n$ are the parameterized 1-qubit rotation operations on distinct qubits. Then, the concept class $\mathcal{H}$ has SQ dimension $\text{SQ-DIM}_{\mathcal{D}}(\mathcal{H}) \geq 3^n$ under any distribution of states that forms a 2-design.

Proof. The simple proof of this proposition relies on the fact that all Pauli operators are pairwise orthogonal for a 2-design, i.e. given two distinct Pauli operators $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ then $\mathbb{E}_{\rho \sim \mathcal{D}} [\text{Tr}(\boldsymbol{P}_1\rho) \text{Tr}(\boldsymbol{P}_2\rho)] = 0$. Therefore, we simply show that the concept class $\mathcal{H}$ is capable of producing any Pauli string not containing the identity.

To proceed, note that we can rewrite the function class as follows:

$$f(|\psi\rangle; \boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n) = \langle\psi| (\boldsymbol{U}_1^\dagger \boldsymbol{Z}_1 \boldsymbol{U}_1) \otimes (\boldsymbol{U}_2^\dagger \boldsymbol{Z}_2 \boldsymbol{U}_2) \otimes \cdots \otimes (\boldsymbol{U}_n^\dagger \boldsymbol{Z}_n \boldsymbol{U}_n) |\psi\rangle. \tag{A.28}$$

To obtain any arbitrary Pauli string, we simply conjugate the $\boldsymbol{Z}_i$ operator for the i-th qubit by a corresponding operation. If the i-th qubit of a Pauli string is equal to $\boldsymbol{X}$, then we set $\boldsymbol{U}_i = \boldsymbol{H}$ or the Hadamard transform. Similarly, if the i-th qubit of a Pauli string is

equal to $\boldsymbol{Y}$, then we set $\boldsymbol{U}_i = \boldsymbol{H}\sqrt{\boldsymbol{Z}}^\dagger$. By conjugation of the individual 1-qubit operators, we thus can produce any Pauli operator in $\{\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}\}^{\otimes n}$. □

Corollary A.3.5. *By application of Theorem A.3.3, the class of functions defined in Proposition A.3.4 consisting of a single layer of parameterized single qubit unitary gates and a fixed global measurement on n qubits requires $2^{\Omega(n)}$ queries to learn for a query tolerance greater than $3^{-\beta n}$, where $\beta = 1/2 - \Omega(1)$.*

Proposition A.3.6 (SQ dimension for $L = \lceil \log_2 n \rceil$, two-qubit gates, and single Pauli $\boldsymbol{Z}$ measurement)**.** *Given n qubits, let $\mathcal{H}$ be the concept class containing functions $f : \mathbb{C}^{2^n} \to \mathbb{R}$ consisting of $\lceil \log_2 n \rceil$ layers of two-qubit gates followed by a Pauli $\boldsymbol{Z}$ measurement on a single qubit. Then, the concept class $\mathcal{H}$ has SQ dimension $\text{SQ-DIM}_{\mathcal{D}}(\mathcal{H}) \geq 4^n - 1$ under any distribution of inputs that forms a 2-design.*

Proof. We will show that $\mathcal{H}$ is powerful enough to perform any nontrivial Pauli measurement (i.e., any Pauli but the identity) and hence construct at least $4^n - 1$ orthogonal functions. Classically, any parity function can be constructed in $\lceil \log_2 n \rceil$ layers, and we use a similar construction here.

Without loss of generality, assume the Pauli measurement is on the first qubit. Let $\boldsymbol{U}(\theta)$ represent a possible unitary that can be applied using the given hypothesis class, resulting in a final measurement of $\boldsymbol{U}(\theta)^\dagger \boldsymbol{Z}_1 \boldsymbol{U}(\theta)$ on a given input state $|\psi\rangle$. We will show that we can parameterize the circuit such that for any Pauli measurement, $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \otimes \cdots \otimes \boldsymbol{P}_n = \boldsymbol{U}(\theta)^\dagger \boldsymbol{Z}_1 \boldsymbol{U}(\theta)$ where $\boldsymbol{P}_i$ indicates the Pauli operator of qubit i (i.e., $\boldsymbol{P}_i \in \{\boldsymbol{I}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}\}$).

To construct any Pauli operator $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \otimes \cdots \otimes \boldsymbol{P}_n$, we follow the steps below:

1. In the first layer, apply a unitary to each qubit i which maps the computational basis to the basis of the Pauli for qubit i. In more detail, if $\boldsymbol{P}_i = \boldsymbol{I}$ or $\boldsymbol{P}_i = \boldsymbol{Z}$, then apply the identity map to keep the basis the same. If $\boldsymbol{P}_i = \boldsymbol{X}$, then apply the Hadamard transform and if $\boldsymbol{P}_i = \boldsymbol{Y}$ then apply the operation $\boldsymbol{H}\sqrt{\boldsymbol{Z}}^\dagger$.

2. In the l-th layer, apply a specific two qubit gate to qubit pairs $\{1, 2^{l-1}+1\}, \{2(2^{l-1})+1, 3(2^{l-1})+1\}, \{4(2^{l-1})+1, 5(2^{l-1})+1\}, \ldots$. For a layer l and a given pair $\{i, j\}$, apply the following gate:

   - if all of $\boldsymbol{P}_i, \boldsymbol{P}_{i+1}, \ldots, \boldsymbol{P}_{j+2^{l-1}}$ are equal to $\boldsymbol{I}$, then apply the identity.

- if any of $\boldsymbol{P}_i, \boldsymbol{P}_{i+1}, \ldots, \boldsymbol{P}_{j-1}$ are not equal to $\boldsymbol{I}$ and all of $\boldsymbol{P}_j, \boldsymbol{P}_{j+1}, \ldots, \boldsymbol{P}_{j+2^{l-1}}$ are equal to $\boldsymbol{I}$ then apply the identity as well.
- if all of $\boldsymbol{P}_i, \boldsymbol{P}_{i+1}, \ldots, \boldsymbol{P}_{j-1}$ are equal to $\boldsymbol{I}$ and any of $\boldsymbol{P}_j, \boldsymbol{P}_{j+1}, \ldots, \boldsymbol{P}_{j+2^{l-1}}$ are not equal to $\boldsymbol{I}$, then apply a swap gate between qubits i and j.
- otherwise, apply the following 2-qubit gate to i and j which conjugates $\boldsymbol{Z} \otimes \boldsymbol{I}$ to $\boldsymbol{Z} \otimes \boldsymbol{Z}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{A.29}$$

3. repeat step 2 above starting from $l = 1$ to $l = \lceil \log_2 n \rceil$. Measuring the first qubit will measure the corresponding desired Pauli. Note, that the single qubit operations of step 1 and the two-qubit operations of step 2 can be combined into a single 2-qubit gate thus not changing the depth.

Following the steps above, at layer l, the measurement of the first qubit corresponds to the Pauli measurement of the first 2^l qubits. Recursively applying this procedure l layers produces any arbitrary Pauli string. □

Corollary A.3.7. *By application of Theorem A.3.3, the class of functions defined in Proposition A.3.6 consisting of $\lceil \log_2 n \rceil$ two-qubit unitary gates and a fixed measurement on a single qubit requires $2^{\Omega(n)}$ queries to learn for a query tolerance greater than $4^{-\beta n}$, where $\beta = 1/2 - \Omega(1)$.*

Proposition A.3.8 (SQ dimension for L layers, neighboring 2-local gates in one-dimensional lattice and fixed single qubit measurement)**.** *Given n qubits, let $\mathcal{H}$ be the concept class containing functions $f : \mathbb{C}^{2^n} \to \mathbb{R}$ consisting of L layers of 2-qubit unitary operations followed by a Pauli Z measurement on a single qubit (labeled qubit m), i.e. functions of the form*

$$f(|\psi\rangle; \boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L) = \langle\psi| \boldsymbol{W}_1^\dagger \boldsymbol{W}_2^\dagger \cdots \boldsymbol{W}_L^\dagger (\boldsymbol{Z}_m) \boldsymbol{W}_L \cdots \boldsymbol{W}_2 \boldsymbol{W}_1 |\psi\rangle, \tag{A.30}$$

where $|\psi\rangle$ is the input to the function and $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L$ are the unitary operations at each layer consisting of tensor products of 2-local unitary operators acting on neighboring

qubits. Then, the concept class $\mathcal{H}$ has SQ dimension $\text{SQ-DIM}_{\mathcal{D}}(\mathcal{H}) \geq 4^{\min(2L,n)} - 1$ *under any distribution of states that forms a 2-design.*

Proof. Our proof relies on the fact that with L layers, one can conjugate the fixed single qubit measurement on qubit m to produce any $2L$-qubit Pauli on the $2L$ qubits within the reverse light cone of m. We follow a proof outline similar to Proposition A.3.6.

Before we proceed, we assume without loss of generality, that L is odd and the first layer applies a two qubit unitary to qubit m and the preceeding qubit $m-1$. It is straightforward to extend this to the case where L is even. Therefore, qubit m is the L-th qubit in the reverse light cone of qubit m, i.e., the light cone traverses qubits $m-L$ to $m+L-1$. For the steps below, we then index the qubits from $-L$ to $L-1$ so that the numbering is relative to qubit m. To perform a given $2L$ Pauli operator $\boldsymbol{P}_{-L} \otimes \boldsymbol{P}_{-L+1} \otimes \cdots \otimes \boldsymbol{P}_{L-1}$ in the reverse light cone of qubit m, we follow the steps below, many of which are copied from Proposition A.3.6.:

1. In the first layer, apply a unitary to each qubit i which maps the computational basis to the basis of the Pauli for qubit i. In more detail, if $\boldsymbol{P}_i = \boldsymbol{I}$ or $\boldsymbol{P}_i = \boldsymbol{Z}$, then apply the identity map to keep the basis the same. If $\boldsymbol{P}_i = \boldsymbol{X}$, then apply the Hadamard transform and if $\boldsymbol{P}_i = \boldsymbol{Y}$ then apply the operation $\boldsymbol{H}\sqrt{\boldsymbol{Z}}^\dagger$.

2. in the L-th layer, for the 2-qubit unitary acting on qubits 0 and -1, apply the following gate:

   - if all of $\boldsymbol{P}_{-L}, \boldsymbol{P}_{-L+1}, \ldots, \boldsymbol{P}_{-1}$ are equal to $\boldsymbol{I}$ and all of $\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{L-1}$ are equal to $\boldsymbol{I}$, then apply the identity.
   - if any of $\boldsymbol{P}_{-L}, \boldsymbol{P}_{-L+1}, \ldots, \boldsymbol{P}_{-1}$ are not equal to $\boldsymbol{I}$ and and any of $\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_{L-1}$ are not equal to $\boldsymbol{I}$, apply the following 2-qubit gate (CNOT) to i and $i+1$ which conjugates $\boldsymbol{I} \otimes \boldsymbol{Z}$ to $\boldsymbol{Z} \otimes \boldsymbol{Z}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{A.31}$$

   - otherwise, apply the swap operation between qubits 0 and -1.

3. In the l-th layer for any $l \neq L$, apply a specific two qubit gate to neighboring qubit pairs $\{-L+l-1, -L+l\}$ on the edge of the reverse light cone. For simplicity, let $i = -L+1-1$ and apply the following gate to qubit pair $\{i, i+1\}$:

   - if all of $\boldsymbol{P}_{-L}, \boldsymbol{P}_{-L+1}, \ldots, \boldsymbol{P}_i$ are equal to $\boldsymbol{I}$, then apply the identity.
   - if any of $\boldsymbol{P}_{-L}, \boldsymbol{P}_{-L+1}, \ldots, \boldsymbol{P}_i$ are not equal to $\boldsymbol{I}$ and $\boldsymbol{P}_{i+1} = \boldsymbol{I}$, then apply a swap between qubits i and $i+1$.
   - otherwise, apply the following 2-qubit gate (CNOT) to i and $i+1$ which conjugates $\boldsymbol{I} \otimes \boldsymbol{Z}$ to $\boldsymbol{Z} \otimes \boldsymbol{Z}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{A.32}$$

   Similarly, for the other edge of the reverse light cone, we apply the same gates, but in "reverse" logic. Here, we apply a 2-qubit unitary to qubit pair $\{L-l-2, L-l-1\}$. For simplicity, let $i = L-l-2$ and apply the following gate to qubit pair $\{i, i+1\}$:

   - if all of $\boldsymbol{P}_{i+1}, \boldsymbol{P}_{i+2}, \ldots, \boldsymbol{P}_{L-1}$ are equal to $\boldsymbol{I}$, then apply the identity.
   - if any of $\boldsymbol{P}_{i+1}, \boldsymbol{P}_{i+2}, \ldots, \boldsymbol{P}_{L-1}$ are not equal to $\boldsymbol{I}$ and $\boldsymbol{P}_{i+1} = \boldsymbol{I}$, then apply a swap between qubits i and $i+1$.
   - otherwise, apply the following 2-qubit gate to i and $i+1$ which conjugates $\boldsymbol{Z} \otimes \boldsymbol{I}$ to $\boldsymbol{Z} \otimes \boldsymbol{Z}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{A.33}$$

4. repeat step 3 above starting from $l = 1$ to $l = L-1$. Measuring the m-th qubit will measure the corresponding desired Pauli. Note, that the single qubit operations of step 1 and the two-qubit operations of step 2 can be combined into a single 2-qubit gate thus not changing the depth.

□

Corollary A.3.9. *By application of Theorem A.3.3, the class of functions defined in Proposition A.3.8 consisting of L layers of neighboring 2-qubit gates and a fixed measurement on a single qubit requires $2^{\Omega(\min(2L,n))}$ queries to learn for a constant query tolerance that does not depend on L and n.*

The above can be generalized to lower bound the statistical query dimension for circuits of L layers on d-dimensional lattices as we show below. In d-dimensional lattices, since the light cone of a single qubit measurement grows at a rate of L^d for an L layers, we can prove that the statistical query dimension grows as $2^{\Omega\left(\min\left(2L,n^{1/d}\right)^d\right)}$.

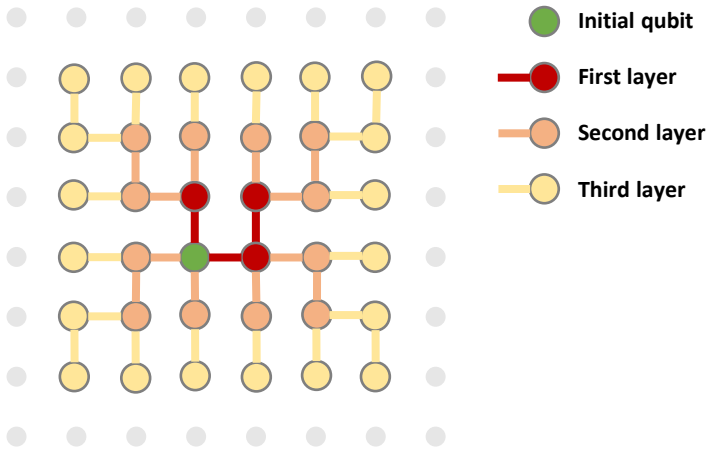


Figure A-1: **Light cone growth.** Growth of the light cone for a 2-dimensional lattice, where the initial qubit is the one that is measured. The size of the lattice grows with the perimeter of the light cone for each layer which consists of local 2-qubit gates applied in each dimension. Each qubit is connected to a qubit in the edge of the light cone of the prior layer, forming a graph which is a tree rooted at the initial qubit.

Proposition A.3.10 (SQ dimension for L layers, neighboring 2-local gates on d-dimensional lattice and fixed single qubit measurement)**.** *Given n qubits, let $\mathcal{H}$ be the concept class containing functions $f : \mathbb{C}^{2^n} \to \mathbb{R}$ consisting of L layers of 2-qubit unitary operations applied in each dimension followed by a Pauli Z measurement on a single qubit (labeled qubit m), i.e. functions of the form*

$$f(|\psi\rangle\,; \boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L) = \langle\psi|\, \boldsymbol{W}_1^\dagger \boldsymbol{W}_2^\dagger \cdots \boldsymbol{W}_L^\dagger\,(\boldsymbol{Z}_m)\, \boldsymbol{W}_L \cdots \boldsymbol{W}_2 \boldsymbol{W}_1\, |\psi\rangle\,, \tag{A.34}$$

where $|\psi\rangle$ is the input to the function and $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L$ are the unitary operations at each layer consisting of tensor products of 2-local unitary operators acting along each dimension on neighboring qubits in a d-dimensional lattice. Then, the concept class $\mathcal{H}$ has SQ dimension $\text{SQ-DIM}_{\mathcal{D}}(\mathcal{H}) = 2^{\Omega\left(\min\left(2L, n^{1/d}\right)^d\right)}$ under any distribution of states that forms a 2-design.

Proof. Our proof relies on the fact that with L layers, one can conjugate the fixed single qubit measurement on qubit m to produce any Pauli on the $\Omega(L^d)$ qubits within the reverse light cone of m. We follow a proof outline similar to Proposition A.3.8.

To be more precise, let us introduce some notation. To perform any Pauli measurement in the reverse light cone at a given layer $l \in [L]$ indexed in reverse order, we apply gates to the perimeter of the reverse light cone at layer $l-1$. We assume there are N_l qubits in the reverse light cone at layer l and index these qubits from 1 to N_l to construct the Pauli $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \otimes \cdots \otimes \boldsymbol{P}_{N_l}$. Like in Proposition A.3.8, we grow the Pauli at each layer.

To grow the light cone and properly choose the 2-qubit gates, we construct a graph which is a tree where the parent of any qubit is the prior qubit which it was connected to in the light cone of the previous layer (see Supplementary Fig. A-1 for an example). The root of the tree is the qubit which is being measured. For example, at layer $l = 1$, the light cone is of size two in each dimension and the qubit being measured is the parent to the child node which it is connected to. To construct any pauli $\boldsymbol{P}_1 \otimes \boldsymbol{P}_2 \otimes \cdots \otimes \boldsymbol{P}_{N_L}$, we follow the steps below:

1. in the l-th layer, for all parent and child qubits p and c respectively connected in the tree at layer l, apply a unitary acting on qubits p and c as follows:

    - if all of the qubits that are descendants of qubit c and qubit c itself have Pauli terms that are equal to $\boldsymbol{I}$, then apply the identity gate between qubit p and c.

    - if any of the qubits that are descendants of qubit c or qubit c itself have Pauli terms that are not equal to $\boldsymbol{I}$ and the Pauli term of qubit p is equal to $\boldsymbol{I}$, then apply the swap gate between p and c.

    - otherwise, apply the following 2-qubit gate to qubits p and c which conjugates

$\boldsymbol{Z} \otimes \boldsymbol{I}$ to $\boldsymbol{Z} \otimes \boldsymbol{Z}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{A.35}$$

2. repeat the step above starting from $l = 1$ to $l = L$.

3. In the first layer ($l = L$), apply a unitary to each qubit i which maps the computational basis to the basis of the Pauli for qubit i. In more detail, if $\boldsymbol{P}_i = \boldsymbol{I}$ or $\boldsymbol{P}_i = \boldsymbol{Z}$, then apply the identity map to keep the basis the same. If $\boldsymbol{P}_i = \boldsymbol{X}$, then apply the Hadamard transform and if $\boldsymbol{P}_i = \boldsymbol{Y}$ then apply the operation $\boldsymbol{H}\sqrt{\boldsymbol{Z}}^\dagger$.

Following the above steps, measuring the single qubit will measure the corresponding desired Pauli. Note, that the single qubit operations of the first layer and the two-qubit operations of that layer can be combined into a single 2-qubit gate thus not changing the depth.

In each layer, 2-qubit gates act along each dimension in some order. We can assume an ordering of the dimensions without loss of generality and assume that we apply gates along that dimension in order. After the first layer, the lattice has size 2 along each dimension. For each layer thereafter, the lattice grows by 2 qubits in each dimension (see Supplementary Fig. A-1 for example). Therefore, the reverse light cone grows at a rate $\Omega(L^d)$. Since the light cone can be at most of size n (number of qubits), then the light cone is of size $2^{\Omega\left(\min\left(2L, n^{1/d}\right)^d\right)}$ for all L layers. □

Corollary A.3.11. *By application of Theorem A.3.3, the class of functions defined in Proposition A.3.10 consisting of L layers of neighboring 2-qubit gates and a fixed measurement on a single qubit requires $2^{\Omega\left(\min\left(2L, n^{1/d}\right)^d\right)}$ queries to learn for a query tolerance that decays no faster than $2^{\omega\left(\min\left(2L, n^{1/d}\right)^d\right)}$. For $L \ll n$, this is equal to $2^{\Omega(L^d)}$ for any constant query tolerance that does not depend on L or n.*

Proposition A.3.12 (SQ dimension for $L = 1$, unitary compiling, and single qubit gates)**.** *Given n qubits, let $\mathcal{H}$ be the concept class containing unitary transformations $\boldsymbol{V} : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ consisting of single qubit rotations in a single layer*

$$\boldsymbol{V}(|\psi\rangle, \boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n) = \boldsymbol{U}_1 \otimes \boldsymbol{U}_2 \otimes \cdots \otimes \boldsymbol{U}_n |\psi\rangle, \tag{A.36}$$

where $|\psi\rangle$ *is the input to the transformation and* $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n$ *are the parameterized* 1*-qubit operations. Then, the concept class* $\mathcal{H}$ *has SQ dimension* $\text{SQ-DIM}_{\mathcal{D}}(\mathcal{H}) \geq 4^n$ *under the* qUSQ *model and any distribution* $\mathcal{D}$ *of inputs that is a* 2*-design.*

Proof. From Lemma A.3.1, we have that $\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\mathcal{D}} = 2^{-n} \Re \left[\text{Tr}(\boldsymbol{U}^\dagger \boldsymbol{V}) \right]$. With one layer of single qubit unitary operations, any Pauli matrix can be constructed. Since $\text{Tr}(\boldsymbol{P}_1 \boldsymbol{P}_2) = 0$ for any two distinct Pauli matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, there are at least 4^n matrices in $\mathcal{H}$ which are orthogonal under the inner product. □

Corollary A.3.13. *By application of Theorem A.3.3, the class of functions defined in Proposition A.3.12 consisting of a single layer of single qubit unitaries requires* $2^{\Omega(n)}$ *queries to learn for a query tolerance greater than* $4^{-\beta n}$*, where* $\beta < 1/2 - \Omega(1)$*.*

A.3.3 Swap Test via Statistical Queries

In the task of unitary compiling, one is given copies of states which are inputs and outputs of a target unitary transformation, and the goal is to learn the unitary transformation from those states. More formally, we aim to learn a unitary $\boldsymbol{U}_*$ given a distribution over inputs or a dataset of m state pairs $\{|\phi_i\rangle, \boldsymbol{U}_* |\phi_i\rangle\}_{i \in [m]}$.

One means of measuring overlaps between states is via the swap test [60]. For pure states, $|\phi\rangle$ and $|\psi\rangle$, the swap test measures the fidelity $|\langle \phi | \psi \rangle|^2$. The measured register in the swap test outputs $|0\rangle$ with probability $1/2 + |\langle \phi | \psi \rangle|^2/2$ and $|1\rangle$ otherwise. As we show in the main text, this quantity can be calculated using queries to qUSQ. We use the helper lemma below to prove this fact.

Lemma A.3.14. *For any distribution* $\mathcal{D}$ *that is a* 2*-design on a Hilbert space of dimension* m*,*

$$\mathbb{E}_{\rho \sim \mathcal{D}} \left[\text{Tr}(\boldsymbol{U}_* \rho \boldsymbol{U}_*^\dagger \boldsymbol{V} \rho \boldsymbol{V}^\dagger) \right] = \frac{m^{-1} \left| \text{Tr}(\boldsymbol{V}^\dagger \boldsymbol{U}_*) \right|^2 + 1}{m + 1}. \tag{A.37}$$

Proof. WLOG, we rewrite the equation above in terms of a distribution over pure states. Since mixed states are themselves distributions over pure states, this can always be done. Therefore, with a slight abuse of notation, we let $\mathcal{D}$ also denote a distribution over unitary

matrices $\boldsymbol{U}$ that forms a 2-design:

$$\begin{aligned}\mathbb{E}_{\rho\sim\mathcal{D}}\left[\mathrm{Tr}(\boldsymbol{U}_*\rho\boldsymbol{U}_*^\dagger\boldsymbol{V}\rho\boldsymbol{V}^\dagger)\right] &= \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}\,|0\rangle\,\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{U}_*^\dagger\boldsymbol{V}\boldsymbol{U}\,|0\rangle\right] \\ &= \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\left|\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}\,|0\rangle\right|^2\right].\end{aligned} \tag{A.38}$$

Using equation A.15 and equation A.16, we have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\left|\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}\,|0\rangle\right|^2\right] &= \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}\,|0\rangle\,\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{U}_*^\dagger\boldsymbol{V}\boldsymbol{U}\,|0\rangle\right] \\ &= \mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\langle I_m^2|\left((\boldsymbol{V}^\dagger\boldsymbol{U}_*)\otimes(\boldsymbol{U}_*^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\otimes\boldsymbol{I}\right)\left(\boldsymbol{U}\otimes\boldsymbol{U}\otimes\bar{\boldsymbol{U}}\otimes\bar{\boldsymbol{U}}\right)|0\rangle^{\otimes 4}\right].\end{aligned} \tag{A.39}$$

Applying equation A.17, we have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{U}\sim\mathcal{D}}\left[\left|\langle 0|\,\boldsymbol{U}^\dagger\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}\,|0\rangle\right|^2\right] &= \frac{1}{m^2-1}\langle I_m^2|\left((\boldsymbol{V}^\dagger\boldsymbol{U}_*)\otimes(\boldsymbol{U}_*^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\otimes\boldsymbol{I}\right)\left(|I_m^2\rangle\langle I_m^2| + |S_m^2\rangle\langle S_m^2|\right)|0\rangle^{\otimes 4} \\ &\quad - \frac{1}{m(m^2-1)}\langle I_m^2|\left((\boldsymbol{V}^\dagger\boldsymbol{U}_*)\otimes(\boldsymbol{U}_*^\dagger\boldsymbol{V})\otimes\boldsymbol{I}\otimes\boldsymbol{I}\right)\left(|I_m^2\rangle\langle S_m^2| + |S_m^2\rangle\langle I_m^2|\right)|0\rangle^{\otimes 4} \\ &= \frac{1}{m^2-1}\left(\mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*]\,\mathrm{Tr}[\boldsymbol{U}_*^\dagger\boldsymbol{V}] + \mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}_*^\dagger\boldsymbol{V}]\right) \\ &\quad - \frac{1}{m(m^2-1)}\left(\mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*]\,\mathrm{Tr}[\boldsymbol{U}_*^\dagger\boldsymbol{V}] + \mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*\boldsymbol{U}_*^\dagger\boldsymbol{V}]\right) \\ &= \left(\frac{1}{m^2-1} - \frac{1}{m(m^2-1)}\right)\left(\left|\mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*]\right|^2 + m\right) \\ &= \frac{m^{-1}\left|\mathrm{Tr}[\boldsymbol{V}^\dagger\boldsymbol{U}_*]\right|^2 + 1}{m+1}.\end{aligned} \tag{A.40}$$

□

A.4 Shallow VQAs as Random Fields

A.4.1 Random Fields on Manifolds

Hardness results from barren plateaus or SQ models both intuitively arise from the exponential decay of quantities necessary to perform optimization. To analyze the shallow circuit setting beyond the SQ model—where such exponentially decaying quantities tend not to exist—we look toward models of variational loss landscapes as random fields on manifolds. This mirrors [92, 82, 20] in studying the loss landscapes of machine learning models via

mapping to certain random fields which are easier to study analytically. As in [20], here we show that certain classes of variational loss functions of shallow quantum models converge in some limit to Wishart hypertoroidal random fields (WHRFs), for which results on the loss landscape are already known [20]. We give a brief review here.

WHRFs in q variables are random fields on a specific tensor product embedding of the hypertorus $\left(S^1\right)^{\times q}$ in $\mathbb{R}^{2^q}$. More specifically, points on this embedding are described by the Kronecker product:

$$\mathbf{w} = \bigotimes_{i=1}^{q} \begin{pmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix} \tag{A.41}$$

for angles $-\pi \leq \theta_i < \pi$. These random fields are then of the form:

$$F_{\text{WHRF}}(\boldsymbol{\theta}) = \mathbf{w}^{\intercal} \cdot \mathbf{J} \cdot \mathbf{w}, \tag{A.42}$$

where $\mathbf{J}$ is drawn from the normalized complex Wishart distribution $\mathcal{CW}_{2^q}(m, \boldsymbol{\Sigma})$ with m degrees of freedom. The complex Wishart distribution is a natural multivariate generalization of the gamma distribution, and is given by the distribution of the square of a complex Gaussian random matrix. Specifically, for $\mathbf{X} \in \mathbb{C}^{N \times m}$ a matrix with i.i.d. complex Gaussian columns with covariance matrix $\boldsymbol{\Sigma}$, the matrix

$$\mathbf{W} = \frac{1}{m} \mathbf{X} \cdot \mathbf{X}^{\dagger} \tag{A.43}$$

is normalized complex Wishart distributed with scale matrix $\boldsymbol{\Sigma}$ and m degrees of freedom. As discussed in [20], the loss landscapes of WHRFs exhibit a complexity phase transition governed by the overparameterization ratio

$$\gamma = \frac{q}{2m}, \tag{A.44}$$

where models with $\gamma \geq 1$ have local minima near the global minimum, and models with $\gamma \ll 1$ have local minima far from the global minimum. Thus, the degrees of freedom parameter m plays a pivotal role in governing the loss landscapes of WHRFs: when q is much smaller than m, training is typically infeasible due to an abundance of "traps" in the training landscape. Our main result here is in demonstrating that even for certain shallow VQAs, the corresponding WHRF is such that $\gamma \ll 1$, and training is infeasible.

A.4.2 Shallow VQAs Converge in Distribution to WHRFs

As discussed informally in the main text, our goal is to demonstrate that certain distributions of shallow variational quantum algorithms (VQAs) weakly converge to Wishart hypertoroidal random fields (WHRFs). The distribution of local minima of WHRFs was shown in [20] to exhibit a phase transition in trainability, where underparameterized models are untrainable due to poor local minima, and overparameterized models exhibit local minima close to the global minimum (though may still be untrainable for other reasons, e.g. due to barren plateaus [246, 74, 256]).

Unlike the nonlocal ansatz case [20], here we are unable to show the full convergence in distribution of shallow local VQAs to WHRFs. Instead, we focus on the joint distribution of the loss function, gradient norm, and Hessian determinant, where the gradient and Hessian have been normalized by the number of parameters q in the reverse light cone of each term in the Pauli expansion of the problem Hamiltonian; by the parameter shift rule [293, 241], it is easy to see that this bounds the gradient norm and Hessian eigenvalues as q is large. The local minima results of [20] depend only on this joint distribution, and thus showing this convergence suffices for our purposes.

We now review the setup of the VQA loss functions we are considering. Throughout the course of this review, we will make various assumptions, particularly on the distribution of gates in the VQA ansatz and on the independence of various reverse light cones; we discuss these assumptions and whether or not they are reasonable in more detail at the end of this Section. As mentioned in the main text, we consider optimizing VQAs on the problem Hamiltonian $\mathbf{H} \neq \mathbf{0}$, which has Pauli decomposition:

$$\mathbf{H} = \sum_{i=1}^{A} \alpha_i \mathbf{P_i}. \tag{A.45}$$

WLOG, we assume here $\mathbf{H}$ is traceless, and that all $\alpha_i > 0$. To simplify our analysis, we will consider the case where the reverse light cone of each term $\alpha_i \mathbf{P_i}$ in the Pauli decomposition of $\mathbf{H}$ is i.i.d. drawn from the same distribution of ansatzes, with the same parameter dependence. To make this more concrete, assume that the reverse light cone of each $\alpha_i \mathbf{P_i}$ is of the form $\mathbf{V_i}(\boldsymbol{\theta})|\mathbf{0}\rangle$ where $\boldsymbol{\theta} \in \mathbb{R}^q$, and has support on a number $l \ll n$ of qubits. In this

regime, we can scale and shift the loss landscape of the standard variational loss function

$$F(\boldsymbol{\theta}) = \langle \boldsymbol{\theta} | \mathbf{H} | \boldsymbol{\theta} \rangle \tag{A.46}$$

to be of the form:

$$F_{\mathrm{VQE}}(\boldsymbol{\theta}) = 1 - \lambda_0^{-1} \sum_{i=1}^{A} \alpha_i \langle \mathbf{0} | \mathbf{V_i}(\boldsymbol{\theta})^\dagger \mathbf{P_i} \mathbf{V_i}(\boldsymbol{\theta}) | \mathbf{0} \rangle = 1 + \|\boldsymbol{\alpha}\|_1^{-1} \sum_{i=1}^{A} \alpha_i \langle \mathbf{0} | \mathbf{V_i}^\dagger(\boldsymbol{\theta}) \mathbf{P_i} \mathbf{V_i}(\boldsymbol{\theta}) | \mathbf{0} \rangle, \tag{A.47}$$

where λ_0 is the ground state energy of $\mathbf{H}$ and $\boldsymbol{\alpha}$ is the vector of all α_i.

We assume that $\mathbf{V_i}(\boldsymbol{\theta})$ is of the form:

$$\mathbf{V_i}(\boldsymbol{\theta}) = \mathbf{W_i}(\boldsymbol{\theta}) \mathbf{U_i}, \tag{A.48}$$

where $\mathbf{U_i}$ are i.i.d. drawn from an ϵ-approximate t-design under the monomial measure on l qubits [161], where $\epsilon = \mathrm{O}(1)$. Note that in particular, though the total ansatz size n may be large, all potential scrambling of the ansatz may only happen locally, in regions of size $l \ll n$; in other words, these ansatzes are not expected to suffer from barren plateaus, particularly if $l = \mathrm{O}(\log(n))$ [74, 256]. $\mathbf{W_i}$ is composed of fixed parameterized rotations which we take WLOG to be of the form $\mathbf{R_{Y_a}}(\theta_b) = \exp(-\mathrm{i}\theta_b \mathbf{Y_a})$ (where as previously mentioned, this parameter dependence is identical across all $\mathbf{W_i}$), fixed gates, and potentially randomly chosen gates such that $\mathbf{W_i}$ itself is a random field. For simplicity, we also assume that all θ_i are independent from one another (i.e. we are in the $r = 1$ regime of [20]), and that each qubit in the reverse light cone has at least one parameterized gate. We also assume that the field $\mathbf{W_i}(\boldsymbol{\theta})$ is rotationally invariant in θ_i.

We now give the formal statement and proof of the loss landscapes of local, shallow VQAs. First, the formal statement:

Theorem A.4.1 (Approximately locally scrambled variational loss functions converge to WHRFs)**.** *Let $p_{VQE,\boldsymbol{\theta}}$ be the joint distribution of the loss function of equation A.47, its gradient norm, and the determinant of its Hessian at $\boldsymbol{\theta}$, where the gradient and Hessian are normalized by q. Let $p_{WHRF,\boldsymbol{\theta}}$ be the same for the WHRF:*

$$F_{WHRF}(\boldsymbol{\theta}) = m^{-1} \sum_{i,j=1}^{2^l} w_i J_{i,j} w_j \tag{A.49}$$

with $m = \frac{\|\boldsymbol{\alpha}\|_1^2}{\|\boldsymbol{\alpha}\|_2^2} 2^{l-1}$ *degrees of freedom, where* $\mathbf{J} \sim \mathcal{CW}_{2^l}(m, \mathbf{I}_{\mathbf{2^l}})$. *Here,* $\mathbf{w}$ *are points on the hypertorus* $\left(S^1\right)^{\times l}$ *parameterized by* $\tilde{\boldsymbol{\theta}}$, *where* $\tilde{\theta}_i$ *is the sum of all* θ_j *on qubit* i. *We then have that* $p_{VQE,\boldsymbol{\theta}}$ *weakly converges to* $p_{WHRF,\boldsymbol{\theta}}$, *up to an error* $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t} + \epsilon + \exp\left(-l\right)\right)\right)$ *in Lévy–Prokhorov distance.*

As we previously mentioned, for technical reasons, we only prove the convergence of the joint distribution of the loss and certain functions of its first two derivatives. We emphasize once more that this does not affect our final conclusions, as all results on the local minima distribution of WHRFs given in [20] depend only on this joint distribution.

To prove Theorem A.4.1, we begin by showing that, up to terms that go to zero polynomially quickly as $\epsilon \to 0, t \to \infty$, one can WLOG consider ansatzes of the form of equation A.47 that are explicitly Haar random within each reverse light cone of size l.

Lemma A.4.2 (Approximate local scrambling bound on the loss function and its derivatives)**.** *Let* $p_{VQE,\boldsymbol{\theta}}$ *be the joint distribution described in Theorem A.4.1. Let* $p_{Haar,\boldsymbol{\theta}}$ *be the same, for* $\mathbf{U_i}$ *taken to be i.i.d. Haar random. We then have that* $p_{VQE,\boldsymbol{\theta}}$ *weakly converges to* $p_{Haar,\boldsymbol{\theta}}$, *up to an error* $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t} + \epsilon\right)\right)$ *in Lévy–Prokhorov distance.*

Proof. Let $\phi_{\mathrm{VQE}}(\mathbf{x} \mid \boldsymbol{\theta})$ be the joint characteristic function of $p_{\mathrm{VQE},\boldsymbol{\theta}}$, and similarly $\phi_{\mathrm{Haar}}(\mathbf{x} \mid \boldsymbol{\theta})$. Since $\mathbf{U_i}$ are assumed to be i.i.d. ϵ-approximate t-designs under the monomial measure, for any moments $M_{\mathrm{VQE},\boldsymbol{\theta}}, M_{\mathrm{Haar},\boldsymbol{\theta}}$ of degree s of $p_{\mathrm{VQE},\boldsymbol{\theta}}, p_{\mathrm{Haar},\boldsymbol{\theta}}$, respectively, we have that:

$$\left|M_{\mathrm{VQE},\boldsymbol{\theta}} - M_{\mathrm{Haar},\boldsymbol{\theta}}\right| = \mathrm{O}\left(\epsilon \mathbf{1}\left[s \leq t\right] + \mathbf{1}\left[s > t\right]\right). \tag{A.50}$$

In particular, for all T sublinear in t,

$$\left|\phi_{\mathrm{VQE}}(\mathbf{x} \mid \boldsymbol{\theta}) - \phi_{\mathrm{Haar}}(\mathbf{x} \mid \boldsymbol{\theta})\right| = \mathrm{O}\left(\epsilon\, \mathrm{poly}\left(T\right) + \frac{(3T)^t}{t!}\right) \tag{A.51}$$

for all $\mathbf{x}$ with $\|\mathbf{x}\|_\infty \leq T$. Similar inequalities hold for the partial derivatives of the joint characteristic functions. Therefore, there exists some $T = \Omega\left(\mathrm{poly}\left(\min\left(t, \frac{1}{\epsilon}\right)\right)\right)$ such that the second bound of Theorem 4 of [348] (with $m = \log\left(T\right)$) on the Lévy–Prokhorov distance is $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t} + \epsilon\right)\right)$. □

Until now, we have considered ansatzes with generic parameter dependence. We now show that up to terms vanishing exponentially quickly in the reverse light cone size l, we

can consider a canonical ansatz form WLOG.

Lemma A.4.3 (Canonical form for Hamiltonian agnostic variational loss functions)**.** *Let $p_{Haar,\boldsymbol{\theta}}$ be the joint distribution described in Lemma A.4.2. Let $p_{can,\boldsymbol{\theta}}$ be the same for the variational loss function*

$$F_{can}(\boldsymbol{\theta}) = \|\boldsymbol{\alpha}\|_1^{-1} \sum_{i=1}^{A} \alpha_i \langle \mathbf{0} | \mathbf{R}(\boldsymbol{\theta})^\dagger \mathbf{U_i}^\dagger \mathbf{P_i} \mathbf{U_i} \mathbf{R}(\boldsymbol{\theta}) | \mathbf{0} \rangle + 1, \tag{A.52}$$

where $\mathbf{R}(\boldsymbol{\theta})$ is the product of the parameterized rotations of equation A.48. We then have that $p_{Haar,\boldsymbol{\theta}}$ weakly converges to $p_{can,\boldsymbol{\theta}}$, up to an error $\tilde{\mathrm{O}}(\mathrm{poly}\exp(-l))$ in Lévy–Prokhorov distance.

Proof. Let us consider (generally mixed) moments involving random variables of the form:

$$K_{ij}(\boldsymbol{\theta_j}) = \langle \mathbf{0} | \mathbf{U_i}^\dagger \mathbf{W_i}(\boldsymbol{\theta_j})^\dagger \mathbf{P_i} \mathbf{W_i}(\boldsymbol{\theta_j}) \mathbf{U_i} | \mathbf{0} \rangle - \langle \mathbf{0} | \tilde{\mathbf{U}}_{\mathbf{ij}}^\dagger \mathbf{U_i}^\dagger \mathbf{W_i}(\boldsymbol{\theta_j})^\dagger \mathbf{P_i} \mathbf{W_i}(\boldsymbol{\theta_j}) \mathbf{U_i} \tilde{\mathbf{U}}_{\mathbf{ij}} | \mathbf{0} \rangle, \tag{A.53}$$

where $\mathbf{U_i}, \tilde{\mathbf{U}}_{\mathbf{ij}}$ are i.i.d. Haar random on l qubits. By the asymptotic free independence of Haar random matrices from constant matrices, and the fact that

$$\mathrm{tr}\left(\mathbf{W_i}(\boldsymbol{\theta_j}) \mathbf{P_i} \mathbf{W_i}(\boldsymbol{\theta_j})^\dagger\right) = \mathrm{tr}\left(|\mathbf{0}\rangle\langle\mathbf{0}| - \tilde{\mathbf{U}}_{\mathbf{ij}} |\mathbf{0}\rangle\langle\mathbf{0}| \tilde{\mathbf{U}}_{\mathbf{ij}}^\dagger\right) = 0, \tag{A.54}$$

we have that any such moment is on the order of $\mathrm{O}(\mathrm{poly}\exp(-l))$ [325]. In particular, it is easy to see that up to an error in Lévy–Prokhorov distance on this order, one can WLOG take $p_{\mathrm{can},\boldsymbol{\theta}}$ as if the gradient and Hessian components had i.i.d. $\mathbf{U_{ij}}$ rather than $\mathbf{U_i}$—for instance, this follows identically to the proof of Lemma A.4.2 with $\epsilon = \mathrm{O}(\mathrm{poly}\exp(-l))$. The result then follows from the unitary invariance of the Haar measure. □

We are now able to prove Theorem A.4.1, following essentially the same procedure as proving Theorem 5 of [20].

Proof. By Lemmas A.4.2 and A.4.3, $p_{\mathrm{VQE},\boldsymbol{\theta}}$ weakly converges to $p_{\mathrm{Haar},\boldsymbol{\theta}}$ up to an error $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t} + \epsilon + \exp(-l)\right)\right)$ in Lévy–Prokhorov distance. By Corollary 1 of [185], this then proves weak convergence of $p_{\mathrm{VQE},\boldsymbol{\theta}}$ to the corresponding joint distribution of a weighted sum of WHRFs each with 2^{l-1} degrees of freedom, up to an additional error in Lévy–Prokhorov

distance exponentially small in l. Weak convergence to $p_{\mathrm{WHRF},\boldsymbol{\theta}}$ then follows from a trivial generalization of Theorem 5 of [20]. □

Scope of results We now comment on the applicability of the results of [20] on the local minima distribution of WHRFs when Theorem A.4.1 holds. All analysis of the local minima distribution of WHRFs in [20] depends only on the joint distribution $p_{\mathrm{WHRF},\boldsymbol{\theta}}$, up to a change in normalization of the gradient and Hessian by l rather than q that does not contribute to the logarithmic asymptotics (i.e. Theorem 7 of [20]) when $q\log(q) = \mathrm{o}(m)$. Thus, in the discussion of the main text, we take this as an extra assumption. Furthermore, we note that the analysis in the main text holds only up to shifts on the order of $\tilde{\mathrm{O}}\left(\mathrm{poly}\left(\frac{1}{t}+\epsilon+\exp(-l)\right)\right)$ in the joint distribution $p_{\mathrm{WHRF},\boldsymbol{\theta}}$, due to the rate of convergence of Theorem A.4.1. However, shifts on this order do not affect the conclusions of [20] for sufficiently large constant ϵ^{-1}, t. For completeness, we summarize this discussion and known results on the loss landscapes of WHRFs [20] with the following Corollary:

Corollary A.4.4 (Shallow, local VQAs have poor loss landscapes)**.** *Let F_{VQE} be a local VQA loss function of the form of equation A.47. Assume all coefficients α_i of the Pauli decomposition of $\mathbf{H}$ are $\Theta(1)$, and*

$$l\log(n) + q\log(q) = \mathrm{o}\left(2^l A\right). \tag{A.55}$$

Then $p_{VQE,\boldsymbol{\theta}}$ weakly converges to $p_{WHRF,\boldsymbol{\theta}}$ as in Theorem A.4.1, where the associated WHRF has a fraction superpolynomially small in n of local minima within any constant additive error of the ground state energy.

Proof. The result follows immediately by applying Theorem 7 of [20] to Theorem A.4.1. □

Assumptions Let us now discuss in more detail the assumptions made in the course of proving Theorem A.4.1. First, we assume that at least some part of the ansatz circuit scrambles some local region around any measured observable; that is, we assume that the ansatz locally is an ϵ-approximate t design for sufficiently large ϵ^{-1}, t. It is known that shallow, local circuits dimensions exhibit this property, when 2-local Haar random gates are applied [161]; thus, in a practical sense, our results assume that the local gates in any distribution of ansatzes under consideration are approximately Haar random. This is a

typical model of Hamiltonian agnostic ansatzes, where the ansatz is chosen independently from the problem Hamiltonian $\mathbf{H}$; see for instance the discussion in the main text and the references therein. The inapplicability of this assumption to Hamiltonian informed ansatzes—particularly for highly symmetric problems—is discussed in more detail in the Discussion of the main text, where we review models that may not suffer from the poor trainability properties we show here.

Our other major assumption is the independence of the $\mathbf{V_i}(\boldsymbol{\theta})$ (up to the repeated use of parameters). Of course, in practice this is almost never true, as otherwise variational optimization would proceed via optimizing each reverse light cone independently. However, given a problem Hamiltonian $\mathbf{H}$ and a shallow ansatz, one can consider a subset of Pauli operators in the Pauli decomposition of $\mathbf{H}$ such that their reverse light cones do not overlap. There is little reason to believe that the loss landscape of this simplified problem should be any more difficult to optimize over than the full problem. We therefore suspect that this assumption is little more than a technical requirement. A similar generalization one could consider is taking the parameters of each $\mathbf{V_i}$ to being almost entirely independent of one another (though not entirely independent, as one could then optimize each subproblem independently, and n would no longer an accurate measure of the size of the problem). However, in this regime we expect the "effective" overparameterization ratio γ to go as $\left(\frac{l}{2^l}\right)^A$, as the problem essentially reduces to simultaneously optimizing A loss functions. For $A \sim n$, for instance, this decays exponentially in n, and thus we believe that models of this form are also not trainable.

A.5 Additional Numerical Experiments

A.5.1 Teacher Student Learning With Checkerboard Ansatzes

One particular challenge with quantum variational learning is that an overparameterized model needs more parameters than the dimension of the quantum input state (exponential in the number of qubits), whereas classically, overparameterization with respect to the size of the data set typically suffices [92, 228, 26]. To illustrate this phenomenon, we consider learning states generated by random shallow checkerboard circuits (denoted the teacher circuit) using checkerboard circuits of the same or more depth (denoted the student circuit). The data set used to train the circuit consist of 512 pairs of inputs randomly drawn from

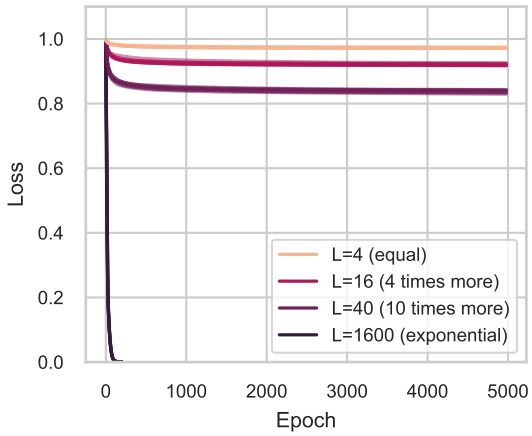


Figure A-2: **Teacher-student performance evaluation for the depth L checkerboard ansatz.** Exponential depth is needed to overparamaterize a model to successfully learn a random circuit of the same form. Here, for each student circuit depth denoted by L, 10 randomly initialized 8 qubit student circuits are trained to learn a random $L = 4$ layer teacher circuit drawn from the same ansatz and parameter distribution.

computational basis states with their corresponding output state taken from applying the input state to the teacher circuit. We use the loss $\ell(|\psi\rangle, |\phi\rangle) = 1 - |\langle\psi\rangle \phi|^2$ to measure the success of learning. Note that, though this is a global loss metric, gradients are analytically calculated to precision sufficient enough to obtain accurate values of the gradients for the relatively small number of qubits considered here.

As shown in Supplementary Fig. A-2, exponential depth (and number of parameters) is needed to always successfully learn the data generated by a shallow checkerboard circuit of 4 layers. We considered ansatzes only over 8 qubits, which is small enough to be able to feasibly overparameterize the models in our simulations. For fewer qubits and shallower circuits, we found that learning with equal numbers of qubits and layers was sometimes successful; but unsurprisingly, as we show in the main text, learning becomes much harder as qubits are added.

A.5.2 Random VQE Model

Here, we empirically analyze the performance of a layer-wise optimizer trained on the random VQE task in the main text. In Supplementary Fig. A-3, we train an 11 qubit ansatz using a layer-wise optimizer [306, 150], which initially trains a single layer of the ansatz and adds layers after every 5000 steps to continually add expressiveness. The target Hamiltonian $\boldsymbol{H}_t$

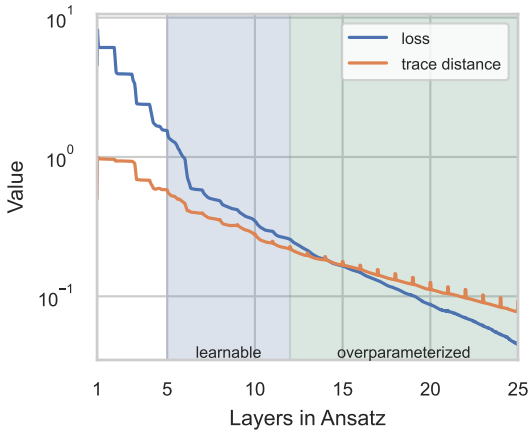


Figure A-3: **Layer-wise learning evaluation.** When optimizing in a layer-wise fashion, the VQE algorithm converges to a local minimum at each layer until the overparameterized regime where the loss function steadily decreases regardless of the number of layers. Even in the learnable regime where the checkerboard ansatz is capable of expressing the global minima, the ansatz is still unable to find the correct parameters for this global minimum. Bumps in the loss function appear due to small instabilities in training immediately after adding a layer.

here has 4 layers of perturbations applied to it. Although layer-wise optimizers can avoid issues with barren plateaus [306], our numerical findings clearly show that this does not guarantee the algorithm will avoid traps in the landscape. After 5 layers, the ansatz has enough parameters to capably express the global optimum (denoted by the label "learnable"), but nevertheless stalls in optimizing to the ground state. Not until there are at least 12 layers, enough to overparameterize the ansatz with respect to the Hilbert space dimension, does learning smoothly converge to the globally optimal solution.

A.5.3 XYZ Hamiltonian Model

All of the numerical experiments performed elsewhere studied settings where the optimization was performed to numerical precision, two qubit gates were fully parameterized, and the existence of a global minimum at zero loss was guaranteed. The analysis there focused on answering the question of whether convergence to the global minimum is empirically likely to be observed. To study a more realistic setting where such favorable conditions cannot be guaranteed but there still exists hope of some good convergence properties, we turn now to the problem of trying to variationally obtain the ground state of an approximately translationally invariant Heisenberg XYZ Hamiltonian [164]. Similar Hamiltonians have been

studied and analyzed in previous works related to VQE [333, 62]. We perform experiments both with and without Gaussian noise added to the gradients to account for shot noise on a quantum computer.

The particular target Hamiltonian we aim to optimize is one where qubits are placed on a 2-dimensional grid and interaction terms take place between neighboring qubits. The Hamiltonian takes the form

$$\boldsymbol{H} = \sum_i \boldsymbol{Z}_i + \sum_{\langle i,j \rangle} \alpha_{ij} \boldsymbol{Z}_i \otimes \boldsymbol{Z}_j + \sum_{\langle i,j \rangle} \beta_{ij} \left(\boldsymbol{X}_i \otimes \boldsymbol{X}_j + 0.66 \boldsymbol{Y}_i \otimes \boldsymbol{Y}_j \right), \tag{A.56}$$

where $\langle i, j \rangle$ sums over the neighboring qubits i and j in the grid and α_{ij} and β_{ij} are random numbers drawn from the normal distribution with standard deviations set to 0.25 and means set to 1 and 3, respectively.

Table A.1: Error in energy from the ground state (normalized by the magnitude of the ground state energy), and trace distance from the ground state, of a VQE optimizing the Heisenberg XYZ model. Results are averaged across 12 random initializations of the experiment for each entry in the table. Note the poor performance of VQE, particularly at the larger problem sizes.

		energy error			trace distance		
	grid size	3×2	5×2	7×2	3×2	5×2	7×2
layers	shots						
3	10000	0.459	0.512	0.504	0.983	0.999	1.000
	400	0.456	0.501	0.392	0.983	0.999	1.000
	inf	0.466	0.512	0.395	0.981	0.999	1.000
9	10000	0.269	0.358	0.351	0.750	0.965	0.998
	400	0.343	0.434	0.386	0.845	0.991	0.994
	inf	0.245	0.350	0.344	0.659	0.924	0.993
15	10000	0.104	0.293	0.303	0.428	0.894	0.997
	400	0.180	0.356	0.318	0.577	0.965	0.987
	inf	0.054	0.244	0.251	0.293	0.842	0.968
21	10000	0.008	0.201	0.214	0.162	0.799	0.982
	400	0.043	0.277	0.247	0.269	0.882	0.984
	inf	0.011	0.178	0.162	0.151	0.747	0.933
27	10000	0.009	0.177	0.200	0.152	0.752	0.976
	400	0.034	0.254	0.251	0.244	0.882	0.976
	inf	0.010	0.122	0.129	0.136	0.663	0.948

As shown in Table A.1, finding the ground state of the XYZ hamiltonian is in general challenging using the ansatz considered. For few layers, the ansatz is not expressible enough to find the target and converges to a poor critical point. For many layers, the VQE algorithm

Table A.2: List of parameter counts, optimizers, and learning rates for the various ansatzes and experiments. L denotes the number of layers and n the number of qubits.

Ansatz	Experiment	# Parameters	Optimizer	Learning Rate
QCNN	Teacher-Student	$16\lceil \log_2 n \rceil = \mathrm{O}(\log n)$	Adam	0.001
Checkerboard	Teacher-Student	$32L\lfloor \frac{n}{2} \rfloor = \mathrm{O}(nL)$	Adam	0.001 (underparameterized) 0.0001 (overparameterized)
	Random VQE (GD)	$128\lfloor \frac{n}{2} \rfloor = \mathrm{O}(n)$	vanilla GD	0.01
	Random VQE (Adam)	$128\lfloor \frac{n}{2} \rfloor = \mathrm{O}(n)$	Adam	0.003
	Adaptive VQE	$160L = \mathrm{O}(L)$	Adam	0.002 (5% reduction each layer)
XYZ ansatz	XYZ Hamiltonian VQE	$7\lfloor \frac{L}{3} \rfloor = \mathrm{O}(L)$	Adam	0.007 (halved every 1000 steps)

tends to converge to a better optimum, but issues with barren plateaus can begin to arise as indicated by the comparison in performance with assuming infinite shots vs. finite shots.

A.6 Details of Numerical Experiments

All experiments were performed in Python using the PyTorch [270] package to perform automatic differentiation. Computation was performed on Nvidia RTX™ A6000 GPUs. Important hyperparameters for the experiments are listed in Table A.2. Unless otherwise stated, all gradients were calculated using analytic formulas for automatic differentation with computer precision (32 bit floating point). Therefore, issues with decaying gradients and barren plateaus do not appear in these simulations for the relatively small number of qubits considered. Gradient based optimization was performed using vanilla gradient descent or the Adam optimizer [212], a popular and effective algorithm for training deep neural networks. We tested other optimizers as well and found no noticeable difference in performance.

Loss surface plot To generate this plot, we chart the loss landscape at initialization of training in the teacher-student setup of the main text for the 14 qubit QCNN circuit. The teacher and student circuit were both initialized as described in Supplementary Note 6.1.

The loss is plotted along two normalized directions of the parameter landscape. Normalization is applied individually to the 3 filters of the 14 qubit QCNN. We loosely follow the "filter-wise" normalization strategy of [224], where we first generate a random direction by drawing a value for each parameter from an i.i.d. standard normal distribution. Then, we divide values for the parameters in a given layer by the Frobenius norm of the matrix for

the corresponding layer.

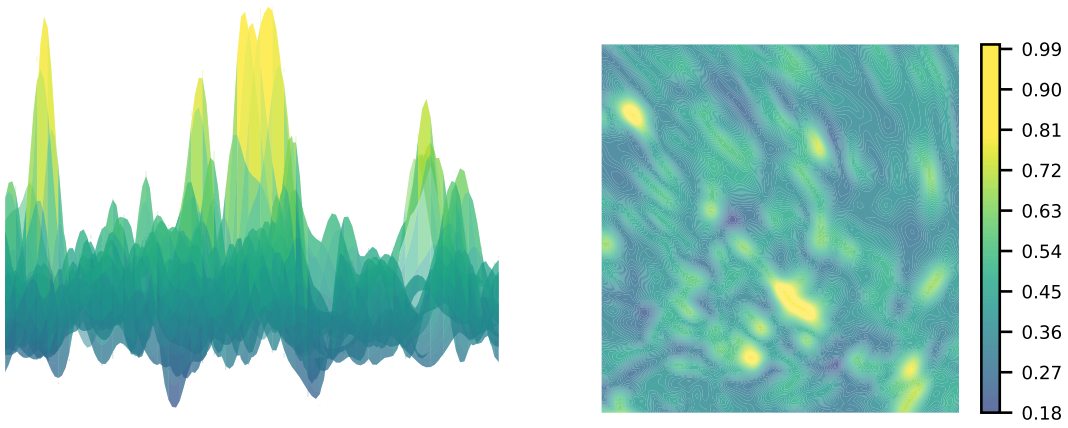


Figure A-4: **Randomly initialized QCNN evaluation.** Loss landscape of the QCNN experiment replicated from the main text, except the initialization of the student circuit is randomly chosen. Here, the global minimum is likely far away and the landscape also appears "bumpy"; all local minima in the region considered here are far from the global optimum.

In the loss surface plot of the main text, we plot the mean squared error loss for the teacher-student task for a batch size of 128 randomly chosen computational basis states. The legend in the plot is shown relative to the maximum value of the loss in the range considered. A value of 0 here corresponds to the loss at the global minimum. The middle of the plot corresponds to the exact parameters of the teacher circuit, and hence, is a global minimum. This setting is, in a sense, an optimistic setting since initialization is near a global minimum. For comparison, we include in Supplementary Fig. A-4 an example of a loss surface where the student circuit is not initialized near the parameters of the teacher circuit. As is evident in this setting, no longer is there a global minimum in the parameter region considered, and the landscape also appears to be filled with traps.

A.6.1 QCNN Experiments

The quantum convolutional neural network (QCNN) is an ansatz originally proposed in [97]. This ansatz features parameter sharing across gates in a single layer. The form of this circuit is provided in Supplementary Fig. A-5. In our experiments, we use the same form of the 2-local ansatz as in [97] and also studied in [275]. Between convolutional layers, we include

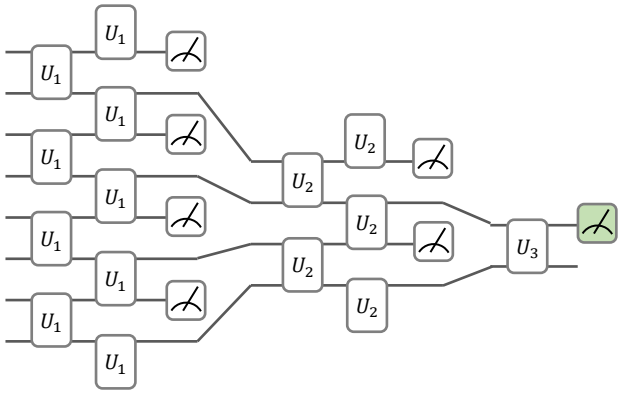


Figure A-5: **QCNN ansatz for 8 qubits.** Layers of shared 2-local unitary transformations are applied followed by measurement of every other qubit. Gates at the edge of the circuit above are applied in a cyclic fashion (i.e. the top and bottom qubit interact). The measurement colored in green is the measurement outcome whose probability we aim to predict in the teacher-student setup. Generically for n qubits, this ansatz has depth $\lceil \log_2 n \rceil$. During training, the 2-local unitaries are fully parameterized for our simulations.

no controlled unitary operations based on the measurement outcomes. In learning settings, we fully parameterize the 2-local unitaries in the skew Hermitian basis of the unitary Lie algebra. To achieve this, we train directly over parameter entries of a matrix $\boldsymbol{M}$ and apply $\mathrm{e}^{\boldsymbol{H}}$, where $\boldsymbol{H} = \boldsymbol{M} - \boldsymbol{M}^\dagger$, to perform the resulting unitary transformation. Entries of the matrix $\boldsymbol{M}$ were initialized i.i.d. from a standard normal distribution.

For the teacher-student experiments in the main text, we aim to predict the outcome of the final green measurement depicted in Supplementary Fig. A-5 for 512 randomly chosen computational basis states. For n qubits, the QCNN ansatz for both the teacher and student circuits have $16\lceil \log_2 n \rceil$ parameters which is a relatively small number compared to the dimension of the Hilbert space. All networks were trained for 5000 epochs and a learning rate of 0.001 using the Adam optimizer.

A.6.2 Checkerboard Ansatz

The checkerboard circuit applies gates in a one-dimensional lattice as shown in Supplementary Fig. A-6. As in the QCNN experiments, we train directly over parameter entries of a matrix $\boldsymbol{M}$ and apply $e^{\boldsymbol{H}}$, where $\boldsymbol{H} = \boldsymbol{M} - \boldsymbol{M}^\dagger$, to perform the resulting unitary transfor-

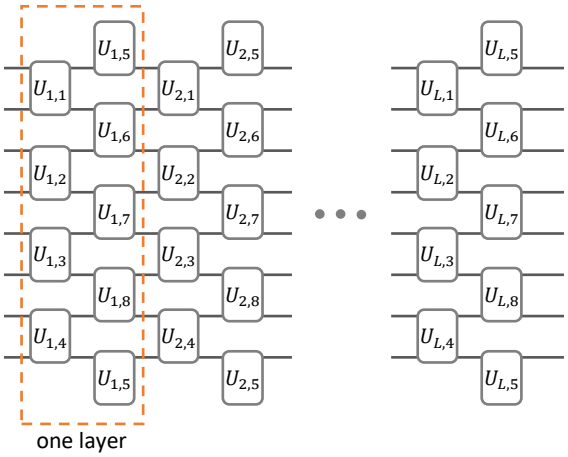


Figure A-6: **Checkerboard ansatz for 8 qubits and L layers.** Gates at the edge of the circuit above are applied in a cyclic fashion (i.e. the top and bottom qubit interact). Generically for n qubits, this ansatz has $32L\lfloor n/2 \rfloor$ parameters. During training, the 2-local unitaries are fully parameterized for our simulations.

mation. Since the exponential map from the Lie algebra is surjective onto the unitary group, this parameterization is capable of expressing any unitary matrix. Entries of the matrix $\boldsymbol{M}$ were initialized i.i.d. from a standard normal distribution.

For the teacher-student simulations of the main text, we train networks over 512 randomly chosen computational basis states which is more than the dimension of the Hilbert space and enough information to recover the full unitary transformation. Optimization was performed using the Adam optimizer and a batch size of 128. Networks were trained for 5000 epochs and training was stopped if the loss fell below 0.001 which only occured for the overparameterized setting. We observed that for fewer than 8 qubits, training was successful with very small probability in the underparameterized setting.

A.6.3 VQE Experiments on Random Hamiltonians

For all of our VQE experiments, the target Hamiltonian $\boldsymbol{H}_t$ was constructed by conjugating a local Hamiltonian of n qubits equal to $\sum_{i=1}^{n} \boldsymbol{Z}_i$ with alternating layers of products of two-qubit unitaries $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. That is, $\boldsymbol{H}_t$ takes the form below as copied from the main text:

$$\boldsymbol{H}_t = \left(\boldsymbol{U}_2^\dagger \boldsymbol{U}_1^\dagger\right)^L \left[\sum_{i=1}^{n} \boldsymbol{Z}_i\right] (\boldsymbol{U}_1 \boldsymbol{U}_2)^L + n\boldsymbol{I}. \tag{A.57}$$

$\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are the tensor product of two-qubit unitaries which for n even take the form:

$$\begin{aligned}\boldsymbol{U}_1 &= \boldsymbol{U}_1^{(1,2)} \otimes \boldsymbol{U}_1^{(3,4)} \otimes \cdots \otimes \boldsymbol{U}_1^{(n-1,n)} \\ \boldsymbol{U}_2 &= \boldsymbol{U}_2^{(2,3)} \otimes \boldsymbol{U}_2^{(4,5)} \otimes \cdots \otimes \boldsymbol{U}_2^{(n,n+1)},\end{aligned} \tag{A.58}$$

where superscripts above indicate the pair of qubits each 2-local unitary acts on and indexing is taken mod n. Each 2 local unitary is drawn from the distribution $e^{\boldsymbol{H}}$, where $\boldsymbol{H} = \boldsymbol{G} - \boldsymbol{G}^\dagger$, and each $\boldsymbol{G}$ is a 4×4 matrix with entries drawn i.i.d. from a random normal distribution. Trained unitaries in the checkerboard ansatz are also initialized in this fashion. Optimization is then performed directly on the entries of the matrix in the Lie algebra which form a complete basis for all of the 2-local unitaries.

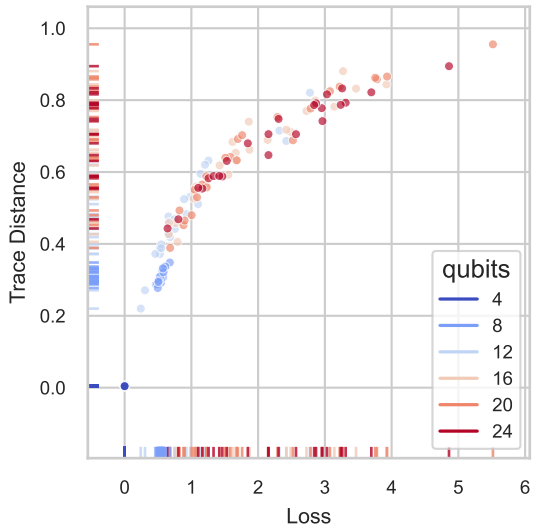


Figure A-7: Adam performance evaluation. Scatter plot showing the values of the loss and trace distance of the final VQE state after 30000 steps of optimization using the Adam optimizer shows that the algorithm converges to poorer local minima as the number of qubits grows. Setting is replicated from the VQE loss experiment from the main text, with the sole change of the optimizer from gradient descent to Adam.

In the VQE loss experiment of the main text, each VQE instance was optimized for 30000 steps using a vanilla gradient descent optimizer with a learning rate of 0.01. For completeness, we replicate this plot with the Adam optimizer in Supplementary Fig. A-7 and unsurprisingly observe similar convergence results. All calculations were performed to computer precision, which provides a best-case setting for optimization via real quantum hardware, since gradients and loss function values would have to be calculated using less

precise sampling methods on actual quantum computers. In layer-wise VQE experiment of the main text, optimization is performed using an adaptive VQE algorithm similar to the one in [150]. Here, a checkerboard ansatz is initialized as a single layer and optimization is performed layer-wise. We set $n = 11$ and small enough such that it is computationally feasible to overparameterize the ansatz. Each 5000 steps of optimization, a layer is added to the ansatz and initialized to the identity mapping. Each additional layer adds 160 trainable parameters to the ansatz. After each layer is added, the learning rate is multiplied by 0.95 to make the training more stable with more parameters. At each point in time, all parameters of the ansatz across all layers are trained. For aesthetic purposes and to see the course of training without significant jumps in the plot, we plot a moving average of the values across 10 sequential datapoints in the main text.

A.6.4 VQE experiments on XYZ Hamiltonian

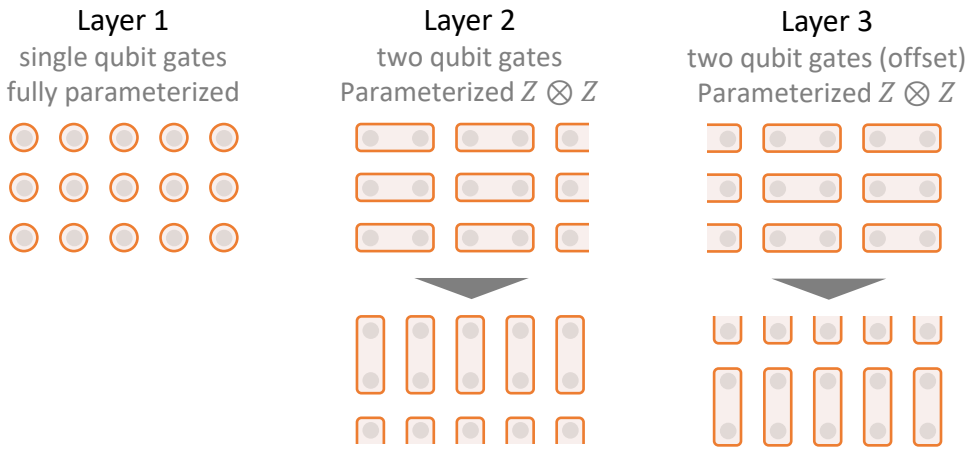


Figure A-8: **Form of the ansatz used for the XYZ Hamiltonian VQE experiments in Supplementary Note 5.3.** Here, alternating blocks of three layers are composed onto each other. The first layer in each block is a fully parameterized single qubit gate. The next two layers are parameterized Pauli $\boldsymbol{Z} \otimes \boldsymbol{Z}$ terms to connect all neighboring qubits. Parameters are shared across a layer to better address the near translationally invariance in the model.

For the XYZ Hamiltonian model empirically analyzed in Supplementary Note 5.3, we implemented a gate-based ansatz which is fully parameterized in the single qubit gates and parameterized only with Pauli $\boldsymbol{Z} \otimes \boldsymbol{Z}$ terms for two qubit gates. The form of the ansatz is depicted in Supplementary Fig. A-8. Since the Hamiltonian of the model is approximately

translationally invariant in both directions, we implemented sharing of parameters across a layer. Parameters were initialized as random normal variables. Each instance was optimized using the Adam optimizer [212] for 5000 steps. The learning rate was initially set to 0.007 and halved every 1000 steps. For calculations of the trace distance to the ground state, the ground state of the Hamiltonian was calculated by performing an eigendecomposition of the complete Hamiltonian. To account for shot noise, random centered Gaussian noise with standard deviation equal to $1/\sqrt{\#\text{ shots}}$ was added to gradients with respect to the parameters.

A.7 Untrainability Beyond Gradient Descent

One may wish to avoid local minima by changing the loss function or performing more advanced versions of gradient-based optimizers. Here, we gives heuristic reasons why these two adjustments will likely not fix any issues of untrainability.

First, we examine changes in the loss function. This is commonly done to avoid barren plateaus and make gradients easier to compute. Let us assume that $\mathcal{L}(\boldsymbol{\theta})$ is our original loss function (as a function of the parameters $\boldsymbol{\theta}$), which is changed to a new loss function $\tilde{\mathcal{L}}(\boldsymbol{\theta})$. Typically, $\tilde{\mathcal{L}}(\boldsymbol{\theta})$ is chosen so that it upper and lower bounds $\mathcal{L}(\boldsymbol{\theta})$, i.e. $C\tilde{\mathcal{L}}(\boldsymbol{\theta}) \leq \mathcal{L}(\boldsymbol{\theta}) \leq D\tilde{\mathcal{L}}(\boldsymbol{\theta})$ for some constants C, D. This guarantees convergence in both metrics when changing the loss function and is the case for e.g. local versions of the inner product and the quantum earth mover's (EM) distance [109, 203]. Now, let us assume that every continuous path from a local minimum at $\boldsymbol{\theta_l}$ to the global minimum $\boldsymbol{\theta^*}$ must increase the loss function by a factor $M > D/C$, i.e. there exists a point in the path that has value at least $M\mathcal{L}(\boldsymbol{\theta_l})$. Then, in the new loss function $\tilde{\mathcal{L}}(\boldsymbol{\theta_l}) \leq \mathcal{L}(\boldsymbol{\theta_l})/C$. Furthermore, at some point in any continuous path, $\mathcal{L}(\boldsymbol{\theta}) > M\mathcal{L}(\boldsymbol{\theta_l})$ which implies that at that point $\tilde{\mathcal{L}}(\boldsymbol{\theta}) \geq \mathcal{L}(\boldsymbol{\theta})/D > M\mathcal{L}(\boldsymbol{\theta_l})/D = \mathcal{L}(\boldsymbol{\theta_l})/C$. Thus, $\boldsymbol{\theta_l}$ is not within a convex region around the global optimum. This may be too restrictive of an assumption since local minima can often be very shallow, but it also seems to be backed by experiments.

Second, we consider changing the optimization algorithm to a second order optimization algorithm such as in [309]. These algorithms perform gradient descent by applying a

transformation to the gradient of the form:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \mu \boldsymbol{\Sigma}^{+} \nabla_{\boldsymbol{\theta_t}} \mathcal{L}(\boldsymbol{\theta_t}), \tag{A.59}$$

where $\boldsymbol{\Sigma}$ incorporates our second order term, e.g. the Hessian or Fubini–Study metric tensor, and $\boldsymbol{\Sigma}^{+}$ is its pseudoinverse. Clearly, in the above, this does not allow one to escape a local minima. Setting $\boldsymbol{\theta_t} = \boldsymbol{\theta}^{*}$ above sets the gradient term to zero, and one again one is stuck in a local minimum.

Though other training methods exist, it is not clear *a priori* why they should succeed. For example, training in a layer-wise fashion also does not work as [203, 306, 65] show. Finally, note that the above methods can be very effective at alleviating barren plateaus. In fact, changes in metric and second order optimization methods are often precisely designed to fix this issue. Nevertheless, these methods only provably converge to the global optimum in convex or close to convex settings, which is not the case for essentially all variational quantum models.

Appendix B

Appendix for Chapter 3

B.1 Related Works

B.1.1 Loss landscape in quantum machine learning

Prior research has theoretically and numerically analyzed the typical properties of loss landscapes in quantum machine learning and control settings. Notably, for commonly used cost functions, prior work has proved and numerically shown the existence of "barren plateaus" characterized by exponentially decaying gradients for large depth quantum parameterized circuits [246, 354, 76], cost functions with global observables [78], and noisy circuits [329]. Furthermore, research in quantum control theory has identified the presence of traps when the control landscape is constrained [253, 272, 216].

Various attempts have been made to potentially avoid these roadblocks. These include methods for initializing circuit parameters [148, 355], algorithms for layer-wise training [305], and choosing ansatzes that can avoid decaying gradients [274, 49]. Despite these efforts, prior work [78] has shown that barren plateaus are unavoidable unless cost functions are appropriately chosen to be local. Thus more pertinent to our work are prior studies that have specifically designed loss metrics or gradient-based algorithms that help avoid issues with barren plateaus. This includes Ref. [78], which considers local operator loss functions, but not necessarily distance metrics. The quantum EM distance, when used as a cost function, is biased towards local operators, gaining the benefits of using local cost functions outlined in [78]. Beyond direct changes to the cost functions, Ref. [180] includes second order derivatives in optimizing the loss function to better navigate flat loss landscapes, and

Ref. [309] constructs an algorithm to optimize over the Fubini-study metric tensor. Though not analyzed here, these methods can be used in tandem with our qWGAN to improve optimization of circuit parameters and increase rates of convergence.

B.1.2 Classical and quantum generative adversarial networks

Generative models, as their name indicates, aim to generate a target object or produce samples from a target distribution by approximating the given target through a learning procedure. In the quantum setting, one popular variant for generative models are Born machines whose cost function is measured by comparing a target classical distribution to the sample distribution of a measurement from a variational quantum circuit [231, 41, 99]. Ref. [41] considers the classical EM distance in their evaluation of the cost function which compares the sampled distribution of the quantum computer to the target distribution.

One commonly used generative algorithm is the generative adversarial network (GAN), a classical algorithm first introduced in [147]. Most relevant to the current work, Ref. [23] constructed the first classical Wasserstein GAN employing an earth mover's distance. Later work improved upon the stability and training of the original Wasserstein GAN by, for example, constructing improved discriminators and generators [155, 352, 251], progressively adding layers during training [191], and employing various regularization techniques [287, 276, 136]. In the classical literature, GANs have been extensively used in many real-world applications [230, 340, 342, 124, 332].

In the quantum setting, quantum GANs were first proposed by Refs. [100, 237]. Simple experiments were performed showing the power of quantum GANs in learning quantum data for relatively small systems that can be simulated or experimentally analyzed [42, 173, 12, 8, 239]. Hybrid classical-quantum GANs, fully classical in the discriminator, generator, and/or loss function, were proposed in Refs. [350, 285, 356, 254]. Ref. [79] proposed a version of a quantum Wasserstein GAN (qWGAN), though the employed earth mover's distance is unitarily invariant (see section B.4 for details). Ref. [167] also proposed a qWGAN structure with a classical discriminator and the classical EM distance as their loss function. Our work differs from both of these prior qWGAN papers in that it implements the first qWGAN with a quantum EM distance. Some early experimental demonstrations of quantum GANs have also been performed on various different systems [174, 12, 308, 173].

B.1.3 Applications of quantum machine learning

Most of the work in quantum machine learning has focused on finding useful applications for quantum machine learning. These include applications in finance [265, 308], chemistry [174], and post-processing quantum outputs [207, 201, 341]. Beyond quantum GANs, there has been a focus in recent years on developing near term quantum algorithms potentially implementable on quantum computers with around 100 qubits. Among the most promising candidates include the quantum approximate optimization algorithm [128, 235, 353], the variational quantum eigensolver [189, 268], and quantum GANs discussed earlier.

B.2 The Classical Earth Mover's Distance

The classical earth mover's (EM) distance, also called Monge-Kantorovich distance, is a distance between probability distributions on a metric space which dates back to Monge [252] and has its roots in the theory of optimal mass transport. Let p, q be probability distributions on the metric space $\mathcal{X}$, which for simplicity we will assume to be finite, and let d be the distance on $\mathcal{X}$. Following the Kantorovich's formulation of the EM distance [190], we define the set of the *couplings* between p and q as the set of the probability distributions on two copies of $\mathcal{X}$ with marginals equal to p and q, respectively. In the interpretation of mass transport, p and q are considered as distributions of a unit amount of mass, and any coupling π prescribes a plan to transform the distribution p into the distribution q, in the sense that $\pi(x, y)$ is the amount of mass that is moved from x to y. Assuming that the cost of moving a unit of mass from x to y is equal to $d(x, y)$, the cost of the coupling π is equal to $\sum_{x,\, y\in\mathcal{X}} \pi(x, y)\, d(x, y)$, *i.e.*, to the expectation value of the distance with respect to π. The EM distance between p and q is given by the minimum cost among all the couplings between p and q. The EM distance has been generalized to a transport cost equal to a power of d, leading to the family of the Wasserstein distances of order α, of which the $\alpha = 1$ case recovers the EM distance. The exploration of the Wasserstein distances has led to the creation of an extremely fruitful field in mathematical analysis, with applications ranging from differential geometry and partial differential equations to machine learning [322, 10, 277, 321].

The EM distance can be considered as a generalization of the total variation distance. Indeed, the EM distance recovers the total variation distance when the distance d on $\mathcal{X}$ is the trivial distance for which all the elements of $\mathcal{X}$ are equivalent, *i.e.*, $d(x, y) = 1$ for any

$x \neq y \in \mathcal{X}$.

When $\mathcal{X}$ is a set of the strings of n bits, the natural choice for d is the Hamming distance, given by the number of different bits. In this case, the EM distance is also known as Ornstein's $\bar{d}$ distance [264].

B.3 Quantum Mechanics and Qubits

Any quantum system has an associated Hilbert space. If the Hilbert space has finite dimension N, it is always isomorphic to $\mathbb{C}^N$. For the sake of simplicity, we restrict our discussion to this case.

We denote a column vector in $\mathbb{C}^N$ with $|\cdot\rangle$, where $\cdot$ is a label for the vector. We will mostly consider vectors with unit norm. For any $|\psi\rangle \in \mathbb{C}^N$, we denote with $\langle\psi| \in \left(\mathbb{C}^N\right)^*$ the row vector whose entries are the complex conjugates of the entries of $|\psi\rangle$. Following the usual rule for matrix multiplication, $\langle\cdot|\cdot\rangle$ denotes the canonical Hermitian inner product of $\mathbb{C}^N$, defined to be antilinear in the first entry and linear in the second.

A *quantum state* is the quantum counterpart of a probability distribution on a set of N elements, and is a positive semidefinite Hermitian matrix in $\mathbb{C}^{N\times N}$ with unit trace. A quantum state is *pure* if it cannot be expressed as a nontrivial convex combination of quantum states. This is the case iff the quantum state is an orthogonal projector with rank one, *i.e.*, if it can be expressed as $|\psi\rangle\langle\psi|$ for some unit vector $|\psi\rangle \in \mathbb{C}^N$. With some abuse of notation, we call also the unit vectors in $\mathbb{C}^N$ quantum states, formally meaning the associated orthogonal projectors. Similarly, we call the inner product between two pure quantum states the inner product between the associated unit vectors. A quantum state is called *mixed* if it is not pure. Two quantum states are called *orthogonal* if the corresponding supports are orthogonal. Any mixed quantum state can be expressed as a convex combination of mutually orthogonal pure quantum states.

An *observable* is the quantum counterpart of a real-valued function on a set of N elements, and is given by an $N \times N$ Hermitian matrix. The expectation value of the observable H on the quantum state ρ is given by $\mathrm{Tr}\left[\rho\, H\right]$.

The Hilbert space associated to a composite quantum system is the tensor product of the Hilbert spaces associated to each subsystem. Let ρ be a quantum state of the composite quantum system with Hilbert space $\mathbb{C}^{N_1} \otimes \mathbb{C}^{N_2}$, *i.e.*, a Hermitian matrix in $\mathbb{C}^{N_1\times N_1} \otimes \mathbb{C}^{N_2\times N_2}$.

We denote with ρ_1 the *marginal* state of ρ on the first subsystem, *i.e.*, the quantum state in $\mathbb{C}^{N_1 \times N_1}$ such that $\mathrm{Tr}\left[\rho_1 H\right] = \mathrm{Tr}\left[\rho \left(H \otimes \mathbb{I}_{N_2}\right)\right]$ for any quantum observable H of $\mathbb{C}^{N_1}$. ρ_1 is equal to the partial trace of ρ over the second subsystem: $\rho_1 = \mathrm{Tr}_2 \rho$.

In this paper, we focus on a quantum system composed of n qubits. A *qubit* is the quantum system associated to the Hilbert space $\mathbb{C}^2$. We denote with $|0\rangle, |1\rangle$ the vectors of its canonical basis, which is also called the computational basis. The Hilbert space of n qubits is $\left(\mathbb{C}^2\right)^{\otimes n}$, and is isomorphic to $\mathbb{C}^N$ with $N = 2^n$. The computational basis of $\left(\mathbb{C}^2\right)^{\otimes n}$ is $\{|x_1\rangle \otimes \ldots \otimes |x_n\rangle : x \in \{0,1\}^n\}$. By the sake of a simpler notation, we denote each vector $|x_1\rangle \otimes \ldots \otimes |x_n\rangle$ with $|x\rangle$, and we set $|0\rangle^{\otimes n} = |0_n\rangle$, $|1\rangle^{\otimes n} = |1_n\rangle$. We denote with $\mathcal{O}_n$ the set of the observables of $\left(\mathbb{C}^2\right)^{\otimes n}$. We say that a linear operator on $\left(\mathbb{C}^2\right)^{\otimes n}$ acts on the i-th qubit if it is equal to a 2×2 matrix acting on the i-th qubit tensored with the identity operator acting on the remaining $n-1$ qubits. The definition of a linear operator acting on a subset of qubits is analogous.

Perhaps the most important observables used and studied in quantum computation are the Pauli matrices. Together with the identity matrix, the Pauli matrices shown below form a basis for the observables on $\mathbb{C}^2$ (*i.e.,* one qubit).

$$\sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{B.1}$$

$$\sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{B.2}$$

$$\sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{B.3}$$

A single Pauli observable can act as a measurement on one qubit; however, multiple qubits can be measured by a "Pauli string" represented by a set of Pauli matrices placed in tensor product form (*e.g.,* $\sigma_X \otimes \sigma_Y \otimes I \otimes \sigma_X$ or equivalently the string 'XYIX').

Pauli operators are often used as parameterized quantum gates. In their parameterized form:

$$R_P(t) = e^{-it\sigma_P/2} = \cos\left(\frac{t}{2}\right) I - i \sin\left(\frac{t}{2}\right) \sigma_P \tag{B.4}$$

where subscript $P \in \{X, Y, Z\}$ indicates the specific Pauli operator chosen.

In our exposition, we outline the computations performed on a quantum computer as

quantum circuits, which are models representing a computation as a sequence of reversible quantum gates and measurement operators. Quantum circuits contain n-bit registers and the sequence of gates are applied accordingly to the qubits in the register. For further details on how to read quantum circuits, the reader is referred to the book [260].

B.4 Quantum Generalizations of Wasserstein Distances

Several quantum generalizations of optimal transport distances have been proposed. One line of research by Carlen, Maas, Datta and Rouzé [66, 67, 68, 288, 102, 318] defines a quantum Wasserstein distance of order 2 from a Riemannian metric on the space of quantum states based on a quantum analog of a differential structure. This quantum Wasserstein distance is intimately linked to both entropy and Fisher information [102], and has led to determine the rate of convergence of the quantum Ornstein-Uhlenbeck semigroup [67, 105]. Exploiting their quantum differential structure, Refs. [288, 68, 135] also define a quantum generalization of the Lipschitz constant and of the earth mover's distance. Alternative definitions of quantum earth mover's distances based on a quantum differential structure are proposed in Refs. [88, 290, 89, 90]. Refs. [6, 4, 182] propose quantum earth mover's distances based on a distance between the vectors of the canonical basis.

Another line of research by Golse, Mouhot, Paul and Caglioti [143, 63, 142, 144, 145, 64] arose in the context of the study of the semiclassical limit of quantum mechanics and defines a family of quantum Wasserstein distances of order 2 built on the notion of couplings. A coupling between the quantum states ρ and σ of $\mathbb{C}^N$ is a quantum state Π of $\left(\mathbb{C}^N\right)^{\otimes 2}$ whose marginal states on the first and on the second subsystems are equal to ρ and σ, respectively. The transport cost of the coupling Π is $\mathrm{Tr}\left[\Pi\, C\right]$, where C is a suitable positive semidefinite $N^2 \times N^2$ cost matrix. Different choices of C will lead to different distances. The square distance between ρ and σ is defined as the minimum cost among all the couplings between ρ and σ. Refs. [143, 63, 142, 144, 145, 64] consider the case of a quantum harmonic oscillator, which is actually infinite dimensional, and choose as cost matrix the quantum analog of the square Euclidean distance:

$$C = (Q_1 - Q_2)^2 + (P_1 - P_2)^2 \ , \tag{B.5}$$

where $Q_{1,2}$ and $P_{1,2}$ are the position and momentum operators of the two subsystems,

respectively. However, the resulting distance has the undesirable property that the distance between a quantum state and itself may not be zero. Ref. [79] notices that the distance between any quantum state and itself is zero whenever the support of the cost matrix C is contained in the antisymmetric subspace with respect to the swap of the two subsystems of $\left(\mathbb{C}^N\right)^{\otimes 2}$. Therefore, Ref. [79] chooses the orthogonal projector onto the antisymmetric subspace as cost matrix, and employs the resulting distance as a cost function for quantum GANs. We stress that this distance is unitarily invariant. Indeed, for any $N \times N$ unitary matrix U, if Π is a coupling between the quantum states ρ and σ, then $U^{\otimes 2}\,\Pi\,U^{\dagger\otimes 2}$ is a coupling between $U\,\rho\,U^\dagger$ and $U\,\sigma\,U^\dagger$, and these two couplings have the same cost since the projector onto the antisymmetric subspace commutes with $U^{\otimes 2}$. Moreover, the only coupling between the pure quantum states $|\psi\rangle$ and $|\phi\rangle$ is the product state $\Pi = |\psi\rangle\langle\psi| \otimes |\phi\rangle\langle\phi|$, whose cost is equal to $\left(1 - |\,\langle\phi|\psi\rangle\,|^2\right)/2$. Therefore, the distance between pure quantum states is a function of their overlap.

Ref. [110] proposes another quantum Wasserstein distance of order 2 based on couplings, with the property that each quantum coupling is associated to a quantum channel. The relation between quantum couplings and quantum channels in the framework of von Neumann algebras has been explored in [123]. The problem of defining a quantum earth mover's distance through quantum couplings has been explored in Ref. [5].

The quantum Wasserstein distance between two quantum states can be defined as the classical Wasserstein distance between the probability distributions of the outcomes of an informationally complete measurement performed on the states, which is a measurement whose probability distribution completely determines the state. This definition has been explored for Gaussian quantum systems with the heterodyne measurement in Refs. [359, 358, 44].

B.5 Toy Model Details

Our toy model (subsection 3.2.1) analyzes the learnability of the GHZ state when using a loss function either corresponding to fidelity (function of inner product between target GHZ state and generated state) or the quantum EM distance. For optimizing over the fidelity, we have a loss function, copied below, that is easily evaluated as a function of $\boldsymbol{\theta}$.

$$F = |\langle GHZ_n|\psi(\boldsymbol{\theta})\rangle|^2 = \left(\frac{\cos(\theta_1)}{\sqrt{2}} + \frac{\prod_{i=1}^{n}\sin(\theta_i)}{\sqrt{2}}\right)^2 \tag{B.6}$$

Here, to perform optimization, we simply perform gradient based updates on the parameters θ_i in the equation above. For all experiments, the Adam optimizer was used to perform gradient updates with a learning rate of 0.2 [212]. For each simulation, 100000 steps of optimization were performed before stopping. Learning is considered successful if $1 - F < 0.02$.

In our toy model, to efficiently approximate the quantum EM distance, we construct a loss function $\tilde{D}_{EM} \approx D_{EM}\left(|\psi(\boldsymbol{\theta})\rangle\langle\psi(\boldsymbol{\theta})|, |GHZ_n\rangle\langle GHZ_n|\right)$ which takes the maximum over $O(n)$ expectations of Pauli operators. We first note that the state $|\psi(\boldsymbol{\theta})\rangle$ is spanned by up to $n+1$ computational basis states.

$$\begin{aligned}|\psi(\boldsymbol{\theta})\rangle &= \cos\theta_1|0_n\rangle + i\sin\theta_1\cos\theta_2|1\rangle|0_{n-1}\rangle - \sin\theta_1\sin\theta_2\cos\theta_3|1_2\rangle|0_{n-2}\rangle + \dots \\ &= \cos\theta_1|0_n\rangle + \sum_{k=1}^{n-1} i^k\left[\prod_{j=1}^{k}\sin\theta_j\right]\cos\theta_{k+1}|1_k\rangle|0_{n-k}\rangle + i^n\left[\prod_{j=1}^{n}\sin\theta_j\right]|1_n\rangle\end{aligned} \tag{B.7}$$

We can write the state above in a vector of length $n+1$ only including the terms in the above span:

$$|\psi(\boldsymbol{\theta})\rangle = \begin{bmatrix}\cos\theta_1 \\ i\sin\theta_1\cos\theta_2 \\ \vdots \\ i^n\prod_{j=1}^{n}\sin\theta_j\end{bmatrix}, \tag{B.8}$$

where the above vector can be easily stored in the memory of a classical computer.

To calculate $\tilde{D}_{EM}$, we first measure the expectation of $|\psi(\boldsymbol{\theta})\rangle$ with respect to the following $2n$ Pauli operators P_i:

$$\begin{aligned}P_i \in \{&\sigma_Z^{(1)}, \sigma_Z^{(2)}, \dots, \sigma_Z^{(n)}, \\ &\sigma_Y^{(1)}, \sigma_X^{(1)}\otimes\sigma_X^{(2)}, \sigma_X^{(1)}\otimes\sigma_X^{(2)}\otimes\sigma_Y^{(3)}, \dots, \sigma_X^{(1)}\otimes\sigma_X^{(2)}\otimes\cdots\otimes\sigma_X^{(n)}\},\end{aligned} \tag{B.9}$$

which is equivalent to the complete set of single qubit Pauli Z operators combined with a multi-qubit Pauli operator for each qubit k consisting of the Pauli X operator or Pauli Y

operator acting on qubit k if k is even or odd respectively (to handle relative phases) and Pauli X operators acting on all qubits $j < k$. In the above, we use the notation $\sigma_L^{(i)}$ to indicate Pauli $L \in \{X, Y, Z\}$ acting on qubit i.

Since as mentioned earlier, $|\psi(\boldsymbol{\theta})\rangle$ is written compactly in vector form, expectations for each of the above operators can be efficiently evaluated using a classical computer. As discussed in subsection 3.2.3, an optimal Hamiltonian whose expectation approximates the quantum EM distance can be efficiently constructed as a parameterized sum of the above operators. Since all Pauli Z operators act on individual qubits, $\tilde{D}_{EM}$ can be calculated as the maximum amongst the following $n+1$ parameterized sums of expectations of operators:

$$\begin{aligned}\tilde{D}_{EM} = \max\Big\{ & \big|\mathbb{E}[\sigma_Z^{(1)}]\big| + \big|\mathbb{E}[\sigma_Z^{(2)}]\big| + \cdots + \big|\mathbb{E}[\sigma_Z^{(n)}]\big|, \\ & \big|\mathbb{E}[\sigma_Y^{(1)}]\big| + \big|\mathbb{E}[\sigma_Z^{(2)}]\big| + \cdots + \big|\mathbb{E}[\sigma_Z^{(n)}]\big|, \\ & \big|\mathbb{E}[\sigma_X^{(1)} \otimes \sigma_Y^{(2)}]\big| + \big|\mathbb{E}[\sigma_Z^{(3)}]\big| + \cdots + \big|\mathbb{E}[\sigma_Z^{(n)}]\big|, \\ & \cdots, \\ & \big|\mathbb{E}[\sigma_X^{(1)} \otimes \sigma_X^{(2)} \otimes \cdots \otimes \sigma_X^{n}]\big| \Big\},\end{aligned} \tag{B.10}$$

where $\mathbb{E}[\cdot]$ indicates the difference in expectation of the operator $\cdot$ on the generated state versus the target GHZ state. For faster simulation, we actually consider the maximum over a simpler set of operators that is equally effective at learning the GHZ state:

$$\begin{aligned}\tilde{D}_{EM} = \max\Big\{ & \big|\mathbb{E}[\sigma_Z^{(1)}]\big| + \big|\mathbb{E}[\sigma_Z^{(2)}]\big| + \cdots + \big|\mathbb{E}[\sigma_Z^{(n)}]\big|, \\ & \big|\mathbb{E}[\sigma_Y^{(1)}]\big|, \\ & \big|\mathbb{E}[\sigma_X^{(1)} \otimes \sigma_Y^{(2)}]\big|, \\ & \cdots, \\ & \big|\mathbb{E}[\sigma_X^{(1)} \otimes \sigma_X^{(2)} \otimes \cdots \otimes \sigma_X^{n}]\big| \Big\}.\end{aligned} \tag{B.11}$$

Using the equation for $\tilde{D}_{EM}$ above, gradient updates can efficiently be performed on the parameters of the circuit. As with the fidelity loss function, we perform optimization with the Adam optimizer at a learning rate of 0.2 [212]. Only up to 10000 steps of optimization were performed since convergence was almost always achieved within about 1000 steps. Learning is considered successful if $|\langle GHZ_n|\psi(\boldsymbol{\theta})\rangle|^2 > 0.98$ after the optimization. Success

was achieved in virtually all instances when using $\tilde{D}_{EM}$.

B.6 Proof of Proposition 3.2.1

For any $k = 0, \ldots, n-1$, let

$$\Delta_k = |\Psi_k\rangle\langle\Psi_k| - |GHZ_n\rangle\langle GHZ_n|\,, \tag{B.12}$$

and let $\mathcal{D}_1$ be the completely dephasing channel acting on the first qubit. From [108, Proposition 3], the quantum EM distance is contractive with respect to a quantum channel acting on a single qubit. We then have on the one hand

$$\|\Delta_k\|_{EM} \geq \|\mathcal{D}_1(\Delta_k)\|_{EM} = \left\| |1_k\rangle\langle 1_k| \otimes \frac{|0_{n-k}\rangle\langle 0_{n-k}| - |1_{n-k}\rangle\langle 1_{n-k}|}{2} \right\|_{EM} = \frac{n-k}{2}\,. \tag{B.13}$$

On the other hand, we have

$$\begin{aligned} \|\Delta_k\|_{EM} &\leq \|\mathcal{D}_1(\Delta_k)\|_{EM} + \|\Delta_k - \mathcal{D}_1(\Delta_k)\|_{EM} = \frac{n-k}{2} + \frac{1}{2}\|\Delta_k - \mathcal{D}_1(\Delta_k)\|_1 \\ &= \frac{n-k}{2} + \frac{1}{2}\begin{cases} 1 & k = 0 \\ \sqrt{2} & k = 1, \ldots, n-1 \end{cases}, \end{aligned} \tag{B.14}$$

where the first equality follows from [108, Proposition 2], stating that $\|X\|_{EM} = \|X\|_1/2$ for any $X \in \mathcal{O}_n$ with $\mathrm{Tr}_1 X = 0$. The claim follows.

B.7 Bias towards local operators

Consider the maximization problem equation 3.20 copied below:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{|W|} c_j w_j \\ \text{subject to} \quad & \sum_{j:i\in\mathcal{I}_j} |w_j| \leq 1, \qquad i = 1, ..., n \end{aligned}$$

Here we prove that the optimal Hamiltonian for the maximization problem contains only terms with few qubits whenever all the coefficients c_j associated to single Pauli operators are $\Omega(1)$.

Proposition B.7.1. *Let $w^* : \{I, X, Y, Z\}^n \to \mathbb{R}$ be the set of parameters that achieve the maximum in equation 3.20, and let*

$$a = \min_{i=1,\ldots,n} \max_{P=X,Y,Z} \left| \mathrm{Tr}\left[(G - \rho_{\mathrm{tar}})\, \sigma_P^{(i)} \right] \right| . \tag{B.15}$$

Then, $w^*_{P_1 \ldots P_n} = 0$ for any $P_1, \ldots, P_n \in \{I, X, Y, Z\}$ such that

$$|c_{P_1 \ldots P_n}| < a\, |\{i = 1, \ldots, n : P_i \neq I\}| . \tag{B.16}$$

In particular, $w^*_{P_1 \ldots P_n} = 0$ for any Pauli string that acts nontrivially on more than $2/a$ qubits.

Proof. The maximization problem equation 3.20 is a linear program with dual

$$\min_{z \in \mathbb{R}^n_{\geq 0}} \sum_{i=1}^{n} z_i \quad : \quad |c_{P_1 \ldots P_n}| \leq \sum_{i \in [n] : P_i \neq I} z_i \quad \forall\, P_1, \ldots, P_n \in \{I, X, Y, Z\} . \tag{B.17}$$

Let $z^* \in \mathbb{R}^n_{\geq 0}$ achieve the minimum in equation B.17. For any $P_1, \ldots, P_n \in \{I, X, Y, Z\}^n$ such that $w^*_{P_1 \ldots P_n} \neq 0$ we have

$$|c_{P_1 \ldots P_n}| = \sum_{i \in [n] : P_i \neq I} z_i^* . \tag{B.18}$$

From equation B.17 we have $a \leq z_i^*$ for any $i = 1, \ldots, n$. Let $P_1, \ldots, P_n \in \{I, X, Y, Z\}$ satisfy equation B.16, and let us assume that $w^*_{P_1 \ldots P_n} \neq 0$. We get from equation B.18

$$|c_{P_1 \ldots P_n}| = \sum_{i \in [n] : P_i \neq I} z_i^* \geq a\, |\{i \in [n] : P_i \neq I\}| , \tag{B.19}$$

which contradicts equation B.16, and the claim follows. □

B.8 Supplementary details of estimated EM distance

Linear relaxation example over product states To help aid intuition for the linear relaxation, consider the simple setting of estimating the quantum EM distance between two n-qubit product states $|\psi_1\rangle$ and $|\psi_2\rangle$ using only single qubit Pauli terms in the linear

relaxation of equation 3.20 copied below:

$$\begin{aligned} &\text{maximize} \quad \sum_{j=1}^{|W|} c_j w_j \\ &\text{subject to} \quad \sum_{j:i\in\mathcal{I}_j} |w_j| \leq 1, \qquad i = 1, ..., n \end{aligned} \tag{B.20}$$

where $c_j = \langle\psi_1| P_j |\psi_1\rangle - \langle\psi_2| P_j |\psi_2\rangle$ and P_j is one of the single qubit Pauli terms that we use in estimating the EM distance. Solving the above outputs an optimal Hamiltonian that takes the form

$$H_{\max} = \sum_{i=0}^{n_{\text{active}}} w_i^* H_i^*, \tag{B.21}$$

where w_i^* and H_i^* are the weights and active operators respectively.

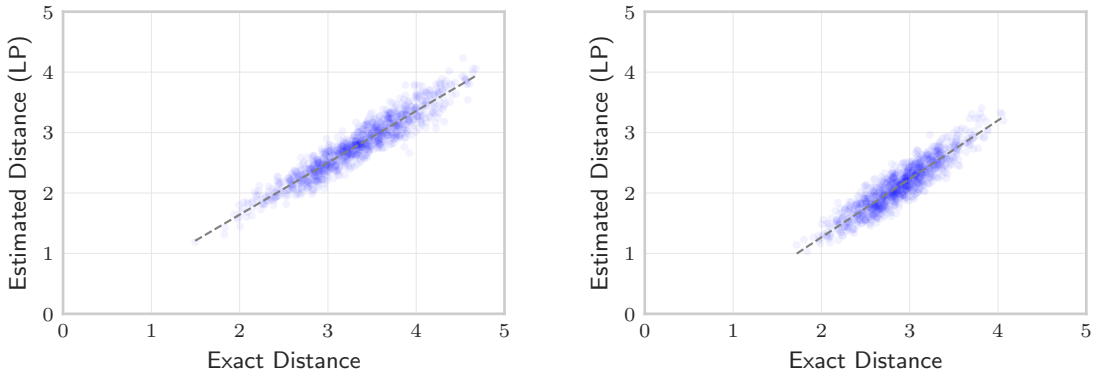
In this setting, for each qubit i, the linear program above will set $w_j^* = \text{sign}(c_j)$ for the c_j with largest magnitude in the set $\{j : i \in \mathcal{I}_j\}$ and $w_j^* = 0$ for all other elements in the set. Thus, the linear program will select the Pauli terms that makes the largest contribution in the difference in expectations between the two states. For example, if $|\psi_1\rangle = |+\rangle|+\rangle$ and $|\psi_2\rangle = |-\rangle|-\rangle$, where $|+\rangle$ and $|-\rangle$ are the $+1$ and -1 eigenvectors of the X basis respectively, then the optimal Hamiltonian unsurprisingly equals $H_{\max} = X \otimes I + I \otimes X$ since these are the single qubit Pauli terms that differ between the two states. Note, that $H_{\max} = X \otimes I + I \otimes X$ also has a Lipschitz constant of one since any single qubit change can only change the value of the Hamiltonian by at most one.

As an aside, adding higher order Pauli terms in the above linear program makes no difference in the outcome. Note, that for any higher order Pauli term, the magnitude of the difference between expectations of the two states $|\psi_1\rangle$ and $|\psi_2\rangle$ will only be captured by differences in their individual qubits. Since the difference in expectation for this higher order Pauli term cannot exceed that of the single qubit Pauli terms within it, these terms will never appear in the optimal Hamiltonian calculated above (see section B.7 for further details).

Correlation of estimated and actual quantum EM distance Exact calculations of the quantum EM distance would require computational resources that grow exponentially with the number of qubits. However, as discussed in the main text, efficient estimates that lower bound the quantum EM distance can be obtained by formulating a new metric $D_{EM}^{(k)}$

optimizing over Hamiltonians of local operators (also see prior section for further motivation for this formalism). This estimated distance can be efficiently calculated as a linear program that requires computational resources that grow polynomially with the number of qubits.

Figure B-1 shows that the exact distance D_{EM} and estimated distance $D^{(2)}_{EM}$ are well correlated for a system of five qubits where calculation of exact distances is possible on a classical computer. In Figure B-1, exact and estimated distances are compared for random product states (Figure B-1a) and states drawn randomly from a depth 3 circuit (Figure B-1b). Though correlations are strong for random product states (Figure B-1a), the estimated distance is not as strong of an approximator for the depth 3 circuit (Figure B-1b) where the slope of the correlation is slightly below one. This situation is one where many of the qubits have interacted with each other and we do not expect learning with the estimated distance to perform well at all times since the approximation is clearly not optimal (*i.e.,* higher order Pauli operators may be needed to approximate the distance better). Nevertheless, the results here lend further support to the proof in the prior section showing that the optimal Hamiltonian in the quantum EM distance is biased towards local operators.



(a) (b)

Figure B-1: Randomly drawn values of the exact distance D_{EM} and estimated distance $D^{(2)}_{EM}$ (calculated from linear program over 2-local Pauli operators) are highly correlated. Individual points are drawn from (a) random product states and (b) states drawn from randomly parameterizing three layers of the barren plateau mixing circuit (Figure B-8). All simulations are for a system of five qubits where exact calculations of the quantum EM distance can be performed efficiently on a classical computer. Each plot contains 1000 randomly drawn data points. A linear fit is plotted over the data as a dotted line.

B.9 Gradients of qWGAN

For the generic version of our generator (equation equation 3.22), optimization is performed over probability parameters p_i and gate parameters in each unitary U_i. The generator optimizes the parameters θ to minimize $\mathrm{Tr}\left[G(\theta)\, H\right]$, where H is the Hamiltonian provided by the discriminator. The following Proposition B.9.1 proves that the gradient of $\mathrm{Tr}\left[G(\theta)\, H\right]$ coincides with the gradient of the EM distance between $G(\theta)$ and ρ_{tar} if H is the optimal Hamiltonian that achieves the EM distance in equation 3.4. Therefore, our learning algorithm decreases the EM distance between $G(\theta)$ and ρ_{tar}. The proof is in subsection B.9.1.

Proposition B.9.1. *For any target quantum state σ, any parametric family of quantum states $\rho(t)$, $0 \leq t \leq T$ that is differentiable in $t = 0$ and any $k = 1, \ldots, n$,*

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} D_{EM}(\rho(t), \sigma)\bigg|_{t=0} &= \max\left(\mathrm{Tr}\left[\rho'(0)\, H\right] : H \in \mathcal{O}_n\,,\ \|H\|_L \leq 1\,,\ \mathrm{Tr}\left[(\rho(0) - \sigma)\, H\right] = D_{EM}(\rho(0), \sigma)\right)\,, \\
\frac{\mathrm{d}}{\mathrm{d}t} D^{(k)}_{EM}(\rho(t), \sigma)\bigg|_{t=0} &= \max\left(\mathrm{Tr}\left[\rho'(0)\, H\right] : H \in \mathcal{O}^{(k)}_n\,,\ \|H\|_{\tilde{L}} \leq 1\,,\ \mathrm{Tr}\left[(\rho(0) - \sigma)\, H\right] = D^{(k)}_{EM}(\rho(0), \sigma)\right)\,.
\end{aligned} \tag{B.22}$$

If $\rho(t)$ admits a differentiable extension to negative values of t, equation B.22 provides the right derivative of $D_{EM}(\rho(t) - \sigma)$, which can be different from the left derivative if the max *in equation B.22 is nontrivial.*

For parameters p_i, the gradient of $\mathrm{Tr}\left[G(\theta)\, H\right]$ can be evaluated using U_i:

$$\frac{\partial D_{EM}}{\partial p_i} = \mathrm{Tr}(U_i \rho_0 U_i^\dagger H_{\max})\,, \tag{B.23}$$

where $H_{\max}$ is the optimal Hamiltonian outputted by the discriminator (equation equation 3.21). Note, that the above is simply the average measured value of $H_{\max}$ for the quantum state $U_i \rho_0 U_i^\dagger$. For gate parameters, we can use standard techniques [293] for evaluating gradients with respect to gate parameters.

B.9.1 Proof of Proposition B.9.1

We prove the claim for the exact quantum EM distance. The proof for the approximated quantum EM distance is completely analogous.

On the one hand, we have for any H as in equation B.22

$$\liminf_{t\to 0^+} \frac{\|\rho(t)-\sigma\|_{EM} - \|\rho(0)-\sigma\|_{EM}}{t} \geq \liminf_{t\to 0^+} \mathrm{Tr}\left[\frac{\rho(t)-\rho(0)}{t} H\right] = \mathrm{Tr}\left[\rho'(0)\, H\right] . \tag{B.24}$$

On the other hand, for any $0 < t < T$, let $H(t) \in \mathcal{O}_n$ be traceless and such that $\|H(t)\|_L \leq 1$ and $\mathrm{Tr}\left[(\rho(t)-\sigma)\, H(t)\right] = \|\rho(t)-\sigma\|_{EM}$. We have

$$\limsup_{t\to 0^+} \frac{\|\rho(t)-\sigma\|_{EM} - \|\rho(0)-\sigma\|_{EM}}{t} \leq \limsup_{t\to 0^+} \mathrm{Tr}\left[\frac{\rho(t)-\rho(0)}{t} H(t)\right] . \tag{B.25}$$

Let $t_k \downarrow 0$ be a sequence that achieves the lim sup in the right-hand side of equation B.25 and such that

$$\lim_{k\to\infty} H(t_k) = H_0 \in \mathcal{O}_n . \tag{B.26}$$

We have

$$\begin{aligned} \|H_0\|_L &= \lim_{k\to\infty} \|H(t_k)\|_L \leq 1 , \\ \mathrm{Tr}\left[(\rho(0)-\sigma)\, H_0\right] &= \lim_{k\to\infty} \mathrm{Tr}\left[(\rho(t_k)-\sigma)\, H(t_k)\right] = \lim_{k\to\infty} \|\rho(t_k)-\sigma\|_{EM} = \|\rho(0)-\sigma\|_{EM} , \end{aligned} \tag{B.27}$$

and

$$\limsup_{t\to 0^+} \frac{\|\rho(t)-\sigma\|_{EM} - \|\rho(0)-\sigma\|_{EM}}{t} \leq \lim_{k\to\infty} \mathrm{Tr}\left[\frac{\rho(t_k)-\rho(0)}{t_k} H(t_k)\right] = \mathrm{Tr}\left[\rho'(0)\, H_0\right] , \tag{B.28}$$

and the claim follows.

B.10 Additional Simulations and Figures

B.10.1 Learning the GHZ state

The analysis in section 3.4 showed that the qWGAN is especially effective at learning the GHZ state. In addition to the results shown in section 3.4, Figure B-2 shows the typical dynamics of learning the GHZ state of 4, 8, and 12 qubits. In all cases, the GHZ state is learned within 1000 steps of optimization.

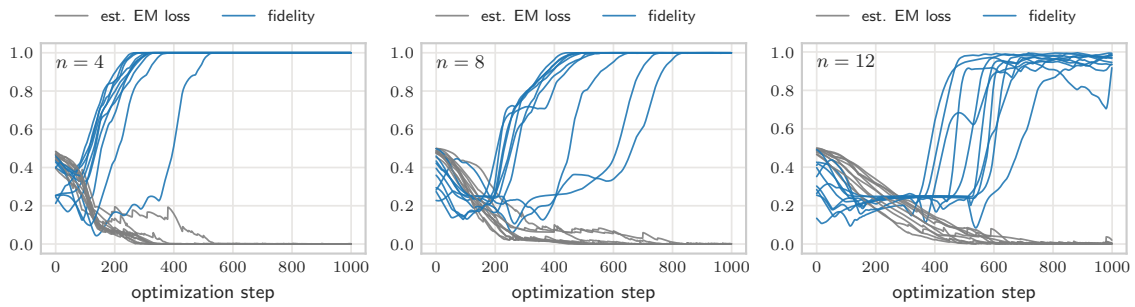


Figure B-2: The qWGAN consistently generates the GHZ state in simulations with circuits of 4, 8, and 12 qubits. Estimated EM loss (quantum EM distance estimated by active operators) is also plotted in above chart, normalized by dividing by the number of qubits. Plots contain one line for each of 10 simulations for each circuit size.

B.10.2 Teacher-student learning

Supplementary to the results in section 3.4, we include Figure B-3 which shows the typical profile of learning in the teacher-student setup. In almost all instances, learning of the state generated by the teacher circuit was achieved.

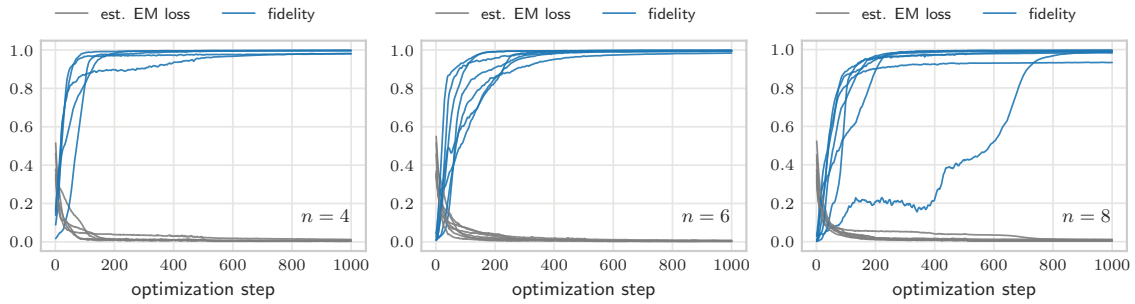


Figure B-3: The student circuit is able to approximate well the state generated by the teacher circuit. Here, the target is constructed by randomly setting the parameters of a depth 2 mixing circuit (teacher circuit). The qWGAN, equipped with a generator circuit of depth 4, successfully learns the target state generated by the teacher circuit. For each plot, 5 simulations are performed.

B.10.3 Gradients of qWGAN vs. conventional GANs

Supplementary to Figure 3-3b, we include further details of the gradients of the quantum EM loss metric and its comparison to a inner product loss metric in Figure B-4. As a reminder, the inner product loss metric is $F = 1 - |\langle \phi_{\text{tar}} | \phi(\boldsymbol{\theta}) \rangle|^2$.

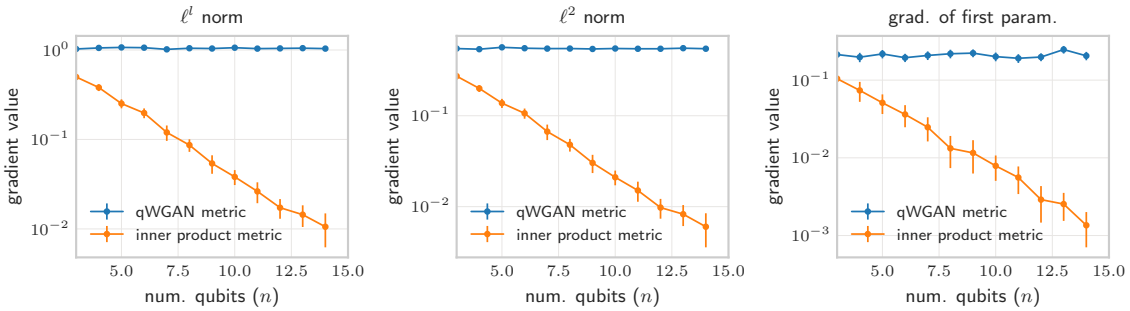


Figure B-4: Comparison of gradients between the quantum EM loss metric and a conventional loss metric that is a function of the inner product. Here the average L1 norm (left), L2 norm (center), and the absolute value of the gradient of the first parameter (right) are shown. Decaying gradients observed in the inner product loss metric. In contrast, regardless of the number of qubits, the gradients of the qWGAN remain stable. Gradients are calculated at first step of optimization. L_1 norm and L_2 norm are divided by n and $\sqrt{n}$ respectively to normalize based on the number of parameters in the circuit. Results are averaged across 100 simulations for each data point

B.10.4 Butterfly circuit learning

In this section, we consider learning the parameters of a "butterfly" circuit which constructs interactions between all qubits in $O(\log_2 n)$ layers. The general form of this circuit is shown in section B.11 and is motivated by prior work in classical machine learning and photonics where similar parameterizations of unitary transformations produced interesting results [245, 186, 101, 93, 301]. Here, the generator takes the form of r_{gen} copies of the parameterized butterfly circuit. The generator aims to learn a target density matrix ρ_{tar} of rank r_{tar} which is generated from a circuit of the same form as the generator but with randomly chosen parameters. In other words,

$$\rho_{\text{tar}} = \frac{1}{r_{\text{tar}}} \sum_{i=1}^{r_{\text{tar}}} U_b(\theta_{\text{ran}}^{(i)}) \rho_0 U_b(\theta_{\text{ran}}^{(i)})^\dagger \tag{B.29}$$

where $U_b(\theta_{\text{ran}}^{(i)})$ is the unitary transformation associated to the butterfly circuit with parameters $\theta_{\text{ran}}^{(i)}$ chosen randomly (we choose each parameter uniformly from $[0, 2\pi)$).

Figure B-5 shows that the qWGAN is effective at learning mixed states of 4 qubits, though learning is clearly more challenging as the rank of the target density matrix increases. We recognize that the form of the generator equation 3.22 may not be well suited to optimization over mixed states. For example, it is often the case that different circuits in the generator optimize to the same critical point in the loss landscape, thus outputting the

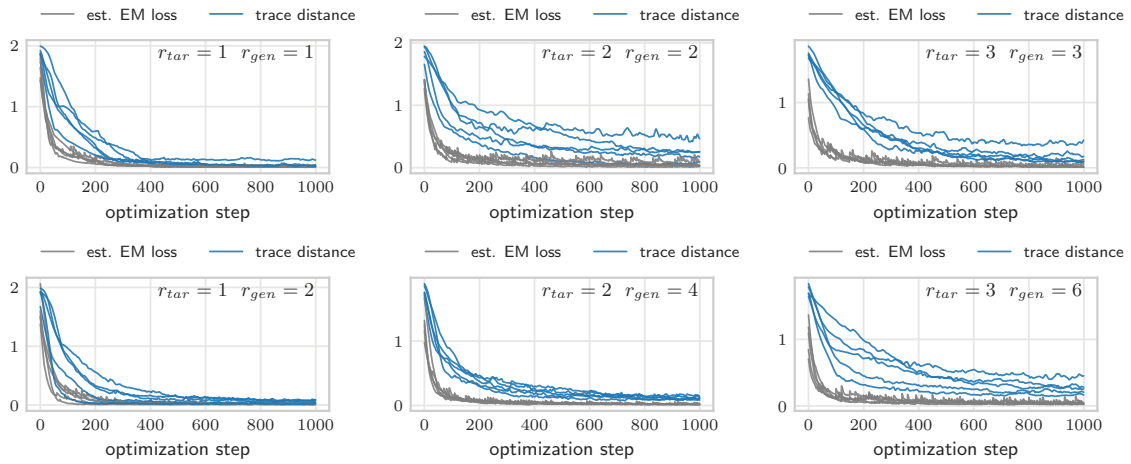


Figure B-5: The qWGAN is also able to generate mixed states that approximate well their given target (also a mixed state) in both trace distance and quantum EM distance. Here, the generator circuit takes the form of a butterfly circuit of 4 qubits (see section B.11), and the target is constructed by randomly setting the parameters of the generator circuit. The qWGAN aims to learn the target density matrix of rank r_{tar} with either $r_{\text{gen}} = r_{\text{tar}}$ or $r_{\text{gen}} = 2r_{\text{tar}}$ parameterized circuits of the same form. For each plot, 5 simulations are performed.

same state. Future improvements to the design of generators can improve the results shown here.

B.10.5 QAOA learning

The quantum approximate optimization algorithm and its related extension to the quantum alternating operator ansatz, both given the acronym QAOA, are promising candidates for achieving quantum speedups in classical optimization problems [130, 157, 128]. Recent work has shown that QAOA is computationally universal [235] and potentially an effective algorithm in a wide range of quantum machine learning settings [206, 320, 331, 353, 171, 80].

Here, we use a QAOA circuit as the generator for our qWGAN to learn the ground state of a simple translationally invariant Ising Hamiltonian cost function C:

$$C = B \sum_{i=1}^{N} \sigma_Z^{(i)} \sigma_Z^{(i+1)}, \tag{B.30}$$

where B is a constant assumed to be positive and $\sigma_Z^{(i)}$ is the Pauli Z operator acting on qubit i. Given the simple translationally invariant form of C, its ground state is spanned by the states $|01\rangle^{\otimes \frac{1}{2}n}$ and $|10\rangle^{\otimes \frac{1}{2}n}$. Note that this setting is different from more traditional

QAOA settings since here we do not aim to find the ground state but instead are given the ground state and aim to construct that state from a parameterized circuit.

For our experiments, we attempt to learn the ground state of C: $\frac{1}{\sqrt{2}}\left(|01\rangle^{\otimes \frac{1}{2}n} + |10\rangle^{\otimes \frac{1}{2}n}\right)$. We use a QAOA circuit which applies, repeating for a depth of L times, a mixing Hamiltonian $e^{-i\alpha_l H_{\text{mix}}}$ and the cost Hamiltonian $e^{-i\beta_l H_C}$ where $l \in \{1, \ldots, L\}$ indicates the layer of the QAOA circuit. In total, the circuit has $2L$ trainable parameters α_l and β_l (see section B.11 for details of circuit).

$$H_{\text{mix}} = \sum_{i=1}^{N} \sigma_X^{(i)} \qquad H_C = \sum_{i=1}^{N} \sigma_Z^{(i)} \sigma_Z^{(i+1)} \tag{B.31}$$

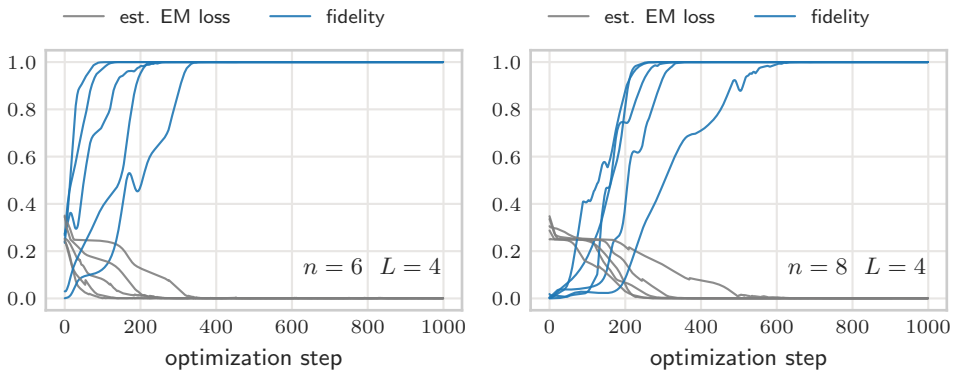


Figure B-6: The qWGAN is effective at learning the ground state of a translationally invariant Ising Hamiltonian. Here, the generator is a QAOA circuit (see section B.11) of depth $L = 4$. Estimated W_1 loss (quantum EM distance estimated by active operators) is also plotted in above chart, normalized by dividing by the number of qubits. For each plot, 5 simulations are performed.

Figure B-6 shows that our qWGAN is very effective at learning the ground state using the QAOA circuit as the generator. Convergence to the ground state is achieved within a few hundred steps of optimization.

B.11 Circuits Used in Experiments

In all our experiments, the generators are parameterized circuits. The form of those circuits are listed below.

- GHZ circuit (section 3.4): circuit is shown in Figure B-7. This circuit differs from that

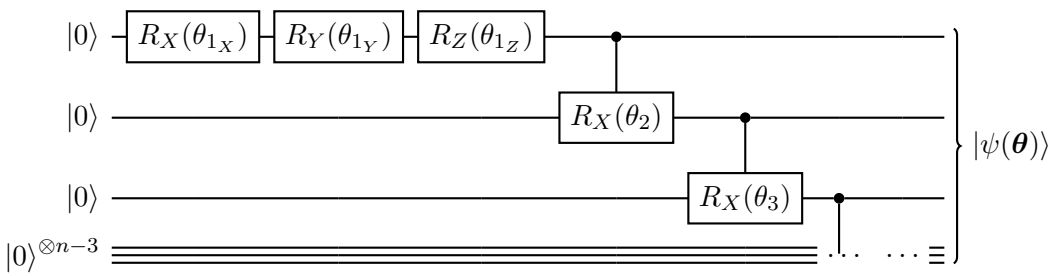


Figure B-7: Circuit for generator in GHZ simulations (section 3.4).

used in the toy model (Figure 3-1a) only in the first qubit. Here, three parameterized Pauli rotations are applied to the first qubit to allow for complete control over the relative phase of the first qubit.

- Mixing circuit (section 3.4): circuit is shown in Figure B-8. This circuit is commonly used in prior literature to show the existence of barren plateaus in the loss landscape [246, 78]. This circuit contains alternating layers of parameterized Pauli Y rotations and pairwise Pauli Z-Z rotations.

- Butterfly circuit (subsection B.10.4): The butterfly circuit takes the form of alternating layers of single qubit Pauli X rotations followed by controlled Pauli X rotations applied in the order of the butterfly pattern (Figure B-9a). The form of the circuit for 4 qubits shown in Figure B-9b.

- QAOA circuit (subsection B.10.5): general form of circuit is shown in Figure B-10a consisting of alternating applications of a mixing Hamiltonian H_{mix} and cost Hamiltonian H_C. An initial layer of Hadamard gates is also included. At a given layer l, Trotterized time evolution circuits are used to apply H_{mix} and H_C for times α_l and β_l respectively [45]. The form of the circuit for 4 qubits and a single QAOA layer ($L = 1$) is shown in Figure B-10b.

B.12 Computational Details

All code used for this paper is available here: https://github.com/bkiani/Quantum-EM-distance-and-qWGAN

Quantum circuit simulations were performed using PennyLane [45] with a backend of Tensorflow [3] or Pytorch [271]. Unless specified otherwise, the Adam optimizer is used for

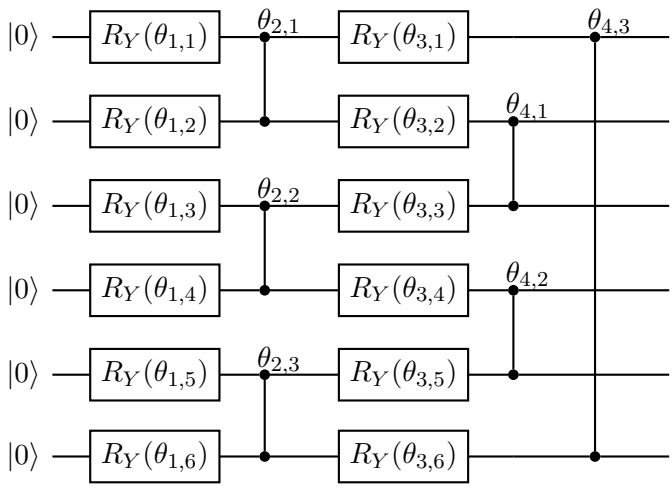
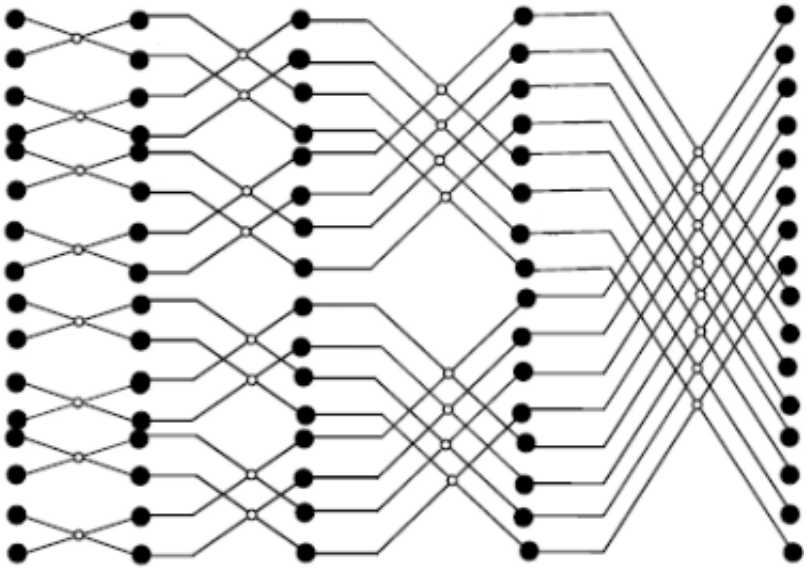


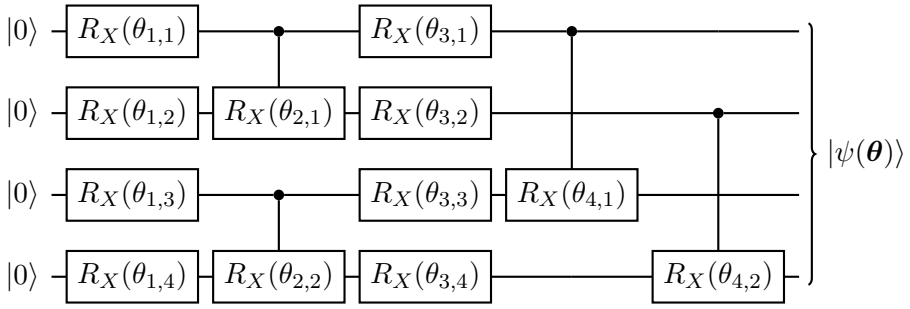
Figure B-8: Single layer of mixing circuit used to perform learning and generate targets in section 3.4. This circuit consists of alternating layers of parameterized Pauli Y rotations and parameterized Pauli Z-Z rotations. The circuit above may be repeated to construct deeper circuits for simulations.

performing gradient-based updates on a generator [212]. The default Adam optimizer was set to a learning rate of 0.01. In some cases, learning was performed in two phases, first with a learning rate of 0.02 decreased to 0.007 for a second phase.

All parameters of the generator are initialized according to a standard normal distribution unless otherwise stated. In its default setting, we cycle the operators of the discriminator every ten optimization steps. When operators are cycled, a cycling threshold of $P = 0.8$ is used (see section 3.3.1). Discriminators are initialized with the set of 2-local Pauli operators.

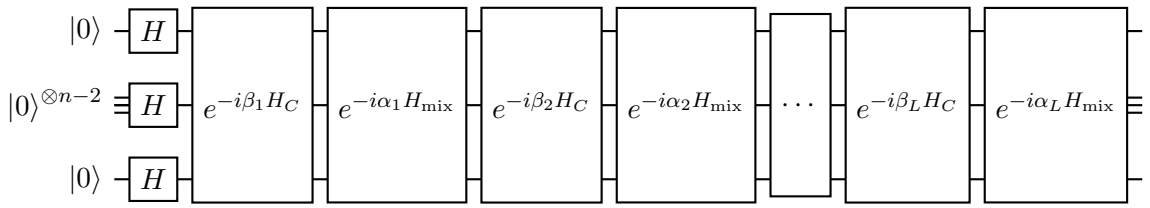


(a)

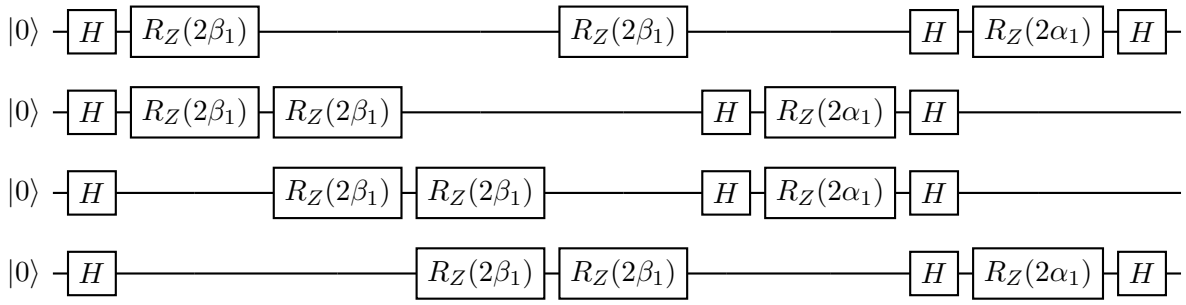


(b)

Figure B-9: (a) Butterfly pattern of interactions, here shown for a system of 16 qubits. (b) Circuit for generator in butterfly circuit simulations (subsection B.10.4) here shown for 4 qubits.



(a)



(b)

Figure B-10: (a) General form for QAOA circuit. (b) QAOA circuit for 4 qubits and $L = 1$.

Appendix C

Appendix for Chapter 4

C.1 The Schur basis

In the presence of permutation invariance, the action of operations can be fully understood by analyzing a much smaller subspace of the larger Hilbert space. To precisely understand the form of that subspace, we turn to the Schur–Weyl decomposition of n qubits into subspaces corresponding to irreducible representations of the symmetric and unitary groups labeled by Young diagrams. Schur–Weyl duality offers a means to perform this decomposition by considering the natural representations of the permutation group and n-fold unitary group acting on n qubits [32, 83]. To describe the Schur basis and the resulting Schur transform, first we note the natural action of a permutation operation $R(\pi)$ acting on qubits:

$$R(\pi)\left|i_1\right\rangle \otimes \left|i_2\right\rangle \otimes \cdots \otimes \left|i_n\right\rangle = \left|i_{\pi^{-1}1}\right\rangle \otimes \left|i_{\pi^{-1}2}\right\rangle \otimes \cdots \otimes \left|i_{\pi^{-1}1}\right\rangle \tag{C.1}$$

as in the main text.

Similarly, a unitary $U \in \mathcal{U}(2)$ acting as the n-fold product $Q(U)$ takes the form

$$Q(U)\left|i_1\right\rangle \otimes \left|i_2\right\rangle \otimes \cdots \otimes \left|i_n\right\rangle = U\left|i_1\right\rangle \otimes U\left|i_2\right\rangle \otimes \cdots \otimes U\left|i_n\right\rangle. \tag{C.2}$$

Schur–Weyl duality takes advantage of the fact that $Q(\cdot)$ and $R(\cdot)$ are each others' commutants, stating that the subspace of $(\mathbb{C}^2)^{\otimes n}$ decomposes as

$$Q(U)R(\pi) \cong \bigoplus_{\lambda} \rho_\lambda(U) \otimes \sigma_\lambda(\pi), \tag{C.3}$$

where λ runs over the set of partitions of n into at most two elements, and $\rho_\lambda(\cdot)$ and $\sigma_\lambda(\cdot)$ are irreducible representations of the unitary group $\mathcal{U}(2)$ and the symmetric group S_n, respectively. Note that irreps of both of these groups are indexed by partitions. More generally, for the space $(\mathbb{C}^d)^{\otimes n}$ of n qudits of dimension d, the λ would span over partitions of n into at most d elements. Partitions can equivalently be enumerated by Young diagrams. For example for the setting of 4 qubits, we have the 3 Young diagrams below that appear in the decomposition above:

$\lambda = (4,0)$:  ,

$\lambda = (3,1)$:  ,

$\lambda = (2,2)$:  .

A consequence of the above is that there exists a basis indexed by $|\lambda, q_\lambda, p_\lambda\rangle$ called the *Schur basis* where the actions of $Q(\cdot)$ and $R(\cdot)$ are separated [32]:

$$Q(U)\,|\lambda, q_\lambda, p_\lambda\rangle = \rho_\lambda(U)\,|\lambda, q_\lambda, p_\lambda\rangle\,, \tag{C.4}$$

$$R(\pi)\,|\lambda, q_\lambda, p_\lambda\rangle = \sigma_\lambda(\pi)\,|\lambda, q_\lambda, p_\lambda\rangle\,, \tag{C.5}$$

where we have implicitly projected onto the subspace indexed by λ. Here, $\rho_\lambda(U)$ and $\sigma_\lambda(\pi)$ act only on the q_λ and p_λ space, respectively. $\rho_\lambda(U)$ and $\sigma_\lambda(\pi)$ are respectively the linear transformations corresponding to the irreducible representations of U_d and S_n for the irreducible representation indexed by λ. The above also presents a useful fact about permutation invariance. Namely, such an operation will act invariantly on the permutation register $|p_\lambda\rangle$ thus significantly reducing the degrees of freedom of a problem. The Schur transform U_{Sch} is a unitary transformation that acts as a change of basis from the computational to the Schur basis described above. The Schur transform can be efficiently implemented on a quantum computer running in time $\mathrm{O}\left(n\mathrm{poly}(d, \log n, 1/\epsilon)\right)$ for error ϵ on qudit systems of

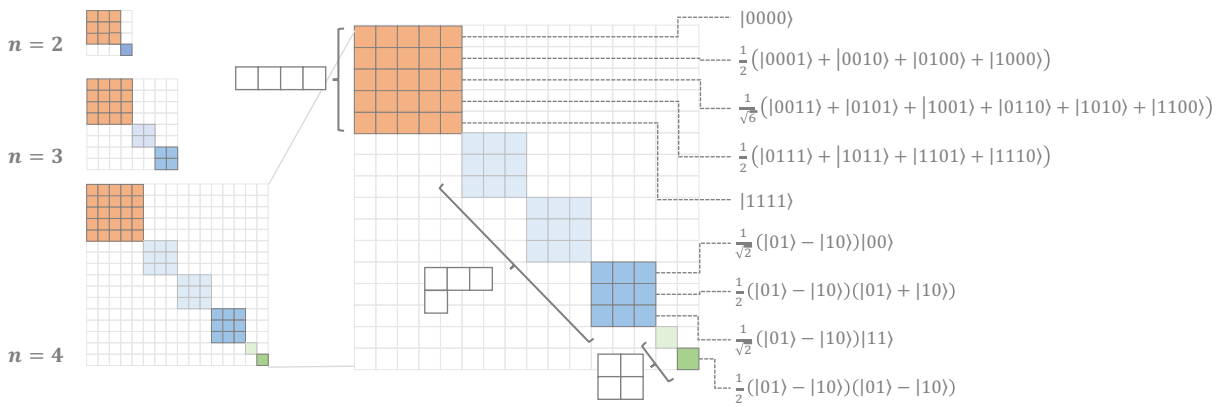


Figure C-1: Graphical depiction of Schur decomposition for $n = 4$ qubits. There are three Young diagrams of at most two rows for 4 qubits. Due to the presence of permutation invariance, we can restrict attention to the darker colored subspaces which correspond to a single subspace over the multiplicity of the permutation irreps. To project onto this darker colored subspace, we use the Young symmetrizer (Eq. equation C.8).

dimension d [32]. We follow the notation of [32]:

$$|\lambda, q_\lambda, p_\lambda\rangle = \sum_{i_1,i_2,\ldots,i_n=0}^{d-1} [U_{Sch}]^{\lambda,q_\lambda,p_\lambda}_{i_1,i_2,\ldots,i_n} |i_1\rangle |i_2\rangle \cdots |i_n\rangle . \tag{C.6}$$

As noted in the main text, the total degrees of freedom reduces to $\binom{n+3}{3}$ in settings with permutation invariance. To see this, note that the dimension of the $|q_\lambda\rangle$ register for a partition (a, b) is equal to $a - b + 1$. Therefore, we have

$$\text{DOF} = \sum_{k=0}^{\lfloor n/2 \rfloor} \left(2k + 1 + n - 2\left\lfloor \frac{n}{2} \right\rfloor\right)^2 = \binom{n+3}{3} \tag{C.7}$$

degrees of freedom. A similar calculation can be performed via a stars-and-bars counting argument. The above is also enumerated by the tetrahedral numbers [262].

To expand and manipulate individual basis states indexed by the $|q_\lambda\rangle$ register, one can use the Young symmetrizer Π_{p_λ} to project onto an explicit basis for each λ [32, 133]. Here, p_λ is a particular Young tableau for the Young diagram λ. The Young symmetrizer projects onto a subspace isomorphic to the subspace spanned by $|q_\lambda\rangle$:

$$\Pi_{p_\lambda} = \frac{\dim(\lambda)}{n!} \left(\sum_{c \in \text{Col}(p_\lambda)} \text{sgn}(c) R(c) \right) \left(\sum_{r \in \text{Row}(p_\lambda)} R(r) \right), \tag{C.8}$$

where $\mathrm{Row}(p_\lambda)$ and $\mathrm{Col}(p_\lambda)$ are the set of permutations which permute integers within only rows and columns of the Young tableau p_λ, respectively [32, 133]. An example of the basis found via application of the Young symmetrizer is shown in Fig. C-1. Throughout our study, we consider the Young tableau formed by filling entries in order first column-wise and then row-wise to be the "canonical" basis that we study. As an example, for 4 qubits, there are the following Young tableaus in our "canonical" basis:

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}, \quad \begin{array}{|c|c|c|}\hline 1 & 3 & 4 \\ \hline 2 \\ \cline{1-1}\end{array}, \quad \begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & 4 \\ \hline\end{array}. \tag{C.9}$$

C.2 Structure coefficients of X for qubit permutation invariance

In this Appendix we evaluate the structure constants of the algebra X of operators symmetric under the action of permutation operators on qubits.

Lemma C.2.1. *The structure coefficients $X_{\boldsymbol{k}}^{\boldsymbol{i},\boldsymbol{j}}$ of the completely symmetrized Pauli representation are given by:*

$$X_{\boldsymbol{k}}^{\boldsymbol{i},\boldsymbol{j}} = \sum_{\substack{\{f_{ab}\}_{a,b\in\{1,x,y,z\}} \\ /Eq.equation\ C.11, Eq.equation\ C.12}} \frac{k_1!}{f_{11}!f_{xx}!f_{yy}!f_{zz}!}\frac{k_x!}{f_{1x}!f_{x1}!f_{yz}!f_{zy}!}\frac{k_y!}{f_{1y}!f_{y1}!f_{xz}!f_{zx}!}\frac{k_z!}{f_{1z}!f_{z1}!f_{xy}!f_{yx}!}(i)^{f_{xy}+f_{yz}+f_z} \tag{C.10}$$

where the variables in the sum are non-negative integers subject to the constraints

$$\sum_{a\in\{1,x,y,z\}} f_{ab} = j_b\ , \qquad \sum_{b\in\{1,x,y,z\}} f_{ab} = i_a, \tag{C.11}$$

and

$$\begin{aligned} f_{11} + f_{xx} + f_{yy} + f_{zz} &=: k_1, \\ f_{1x} + f_{x1} + f_{yz} + f_{zy} &= k_x, \\ f_{1y} + f_{y1} + f_{xz} + f_{zx} &= k_y, \\ f_{1z} + f_{z1} + f_{xy} + f_{yx} &= k_z. \end{aligned} \tag{C.12}$$

Proof. For calculating the structure constants X, we first note that

$$A_{\boldsymbol{j}} = \frac{1}{i_1!i_x!i_y!i_z!} \sum_{\pi \in \mathrm{S}_n} R(\pi) \left(\sigma_1^{\otimes i_1} \otimes \sigma_x^{\otimes i_x} \otimes \sigma_y^{\otimes i_y} \otimes \sigma_z^{\otimes i_z} \right) R^{-1}(\pi) = \sum_{p_{\boldsymbol{i}} \in P_{\boldsymbol{i}}} p_{\boldsymbol{i}} \ , \tag{C.13}$$

where $P_{\boldsymbol{i}}$ is the set of Pauli words with i_a times σ_a for $a \in \{1, x, y, z\}$. Now we evaluate the product:

$$A_{\boldsymbol{i}} \cdot A_{\boldsymbol{j}} = \sum_{p_{\boldsymbol{i}} \in P_{\boldsymbol{i}}, p_{\boldsymbol{j}} \in P_{\boldsymbol{j}}} p_{\boldsymbol{i}} p_{\boldsymbol{j}} = \sum_{\boldsymbol{k}, p_{\boldsymbol{k}} \in P_{\boldsymbol{k}}} \sum_{\substack{p_{\boldsymbol{i}} \in P_{\boldsymbol{i}}, p_{\boldsymbol{j}} \in P_{\boldsymbol{j}}: \\ p_{\boldsymbol{i}} p_{\boldsymbol{j}} = \alpha_{p_{\boldsymbol{i}}, p_{\boldsymbol{j}}, p_{\boldsymbol{k}}} \cdot p_{\boldsymbol{k}}}} \alpha_{p_{\boldsymbol{i}}, p_{\boldsymbol{j}}, p_{\boldsymbol{k}}} \cdot p_{\boldsymbol{k}}. \tag{C.14}$$

This is a sum over products of exponentionally many Pauli words. The idea to evaluate this is that many of the summands have equal value, so it suffices to sum over a few different values multiplied by the number of summands with that value.

For every summand, define the subsets of qubits L_{ab} for $a, b \in \{1, x, y, z\}$,

$$L_{ab} \coloneqq \{l : (p_{\boldsymbol{i}})_l = \sigma_a, (p_{\boldsymbol{j}})_l = \sigma_b, 0 \leq l < n\}, \tag{C.15}$$

and let

$$f_{ab} \coloneqq |L_{ab}| \tag{C.16}$$

be the numbers of elements in those subsets. Since every Pauli operator i_l at a qubit l is paired with some other Pauli operator j_l, f_{ab} fulfill the constraints in Eq. equation C.11. The multiplication algebra of Pauli operators directly implies Eq. equation C.12.

Let us now count how many Pauli words there are in the sum for a fixed set of numbers f_{ab} and a fixed resulting Pauli word $p_{\boldsymbol{k}}$. Every triple $p_{\boldsymbol{i}}$, $p_{\boldsymbol{j}}$, $p_{\boldsymbol{k}}$ corresponds to a decomposition of each $\boldsymbol{k}_c$-element set of qubits $\{l : (p_{\boldsymbol{k}})_l = \sigma_c\}$ for $c \in \{1, x, y, z\}$ into four subsets L_{ab} for the four different combinations $a, b \in \{1, x, y, z\}$ with $\sigma_a \sigma_b \propto \sigma_c$ under the Pauli algebra. For each c, the number of decompositions into the corresponding four subsets is given by

$$\frac{\boldsymbol{k}_c!}{\prod_{a,b:\sigma_a\sigma_b \propto \sigma_c} f_{ab}!} \ . \tag{C.17}$$

In total, the number of decompositions into four subsets for different c is given by

$$\frac{k_1!}{f_{11}!f_{xx}!f_{yy}!f_{zz}!} \frac{k_x!}{f_{1x}!f_{x1}!f_{yz}!f_{zy}!} \frac{k_y!}{f_{1y}!f_{y1}!f_{xz}!f_{zx}!} \frac{k_z!}{f_{1z}!f_{z1}!f_{xy}!f_{yx}!}. \tag{C.18}$$

Finally, the prefactor $\alpha_{p_{\boldsymbol{i}},p_{\boldsymbol{j}},p_{\boldsymbol{k}}}$ in Eq. equation C.14 only depends on the f_{ab}. Using the Pauli algebra,

$$\begin{aligned} &\sigma_x\sigma_y = i\sigma_z \quad \sigma_y\sigma_z = i\sigma_x \quad \sigma_z\sigma_x = i\sigma_y \\ &\sigma_y\sigma_x = -i\sigma_z \quad \sigma_z\sigma_y = -i\sigma_x \quad \sigma_x\sigma_z = -i\sigma_y \ , \end{aligned} \tag{C.19}$$

it is given by

$$\alpha_{p_{\boldsymbol{i}},p_{\boldsymbol{j}},p_{\boldsymbol{k}}} = (i)^{f_{xy}+f_{yz}+f_{zx}}(-i)^{f_{yx}+f_{xz}+f_{zy}} \ . \tag{C.20}$$

Using Eq. equation C.18 and Eq. equation C.20 in Eq. equation C.14 directly yields Eq. equation C.10. □

Let us quickly discuss the complexity of the computation of $X_{\boldsymbol{k}}^{\boldsymbol{i},\boldsymbol{j}}$. In the summation of Eq. equation C.10, we sum over 16 variables within a range of the order n, so if we naively evaluate the sum, we already obtain a polynomial runtime $\mathrm{O}(n^{16})$. However, due to the constraint Eq. equation C.11, we can reduce the summation to only 9 variables f_{ab} with $a, b \in \{x, y, z\}$. Eq. equation C.12 poses another three independent constraints, reducing the summation to 6 variables. Thus, an individual entry $X_{\boldsymbol{k}}^{\boldsymbol{i},\boldsymbol{j}}$ can be calculated in $\mathrm{O}(n^6)$ runtime, whereas all $\mathrm{O}(n^9)$ coefficients together take runtime $\mathrm{O}(n^{15})$.

Note that this is only the runtime for a naive evaluation of the sum in Eq. equation C.10. It seems likely that the runtime $\mathrm{O}(n^6)$ for the evaluation of a single coefficient can be reduced to a smaller exponent. We will leave this open to further investigation.

C.3 Irrep basis of A for qubit permutation invariance

In this section, we compute the matrix elements $F_{q_\lambda,q'_\lambda}^{\boldsymbol{i},\lambda}$ from the main text for the case of S_n action on n qubits by permutation. To this end, we first find the irrep basis $|\lambda, q_\lambda, p_{\lambda 0}\rangle$ where $p_{\lambda 0}$ is a standard choice of Young tableau, and then consider the representation A in this basis.

Lemma C.3.1. *Recalling that λ is given by a Young diagram, we choose $p_{\lambda 0}$ to be the standard Young tableaux for that diagram, with numbers increasing first in the column direction and then in row direction, as shown in Eq. equation C.9. Then the tensor components $F_{q_\lambda,q'_\lambda}^{\boldsymbol{i},\lambda}$*

discussed in the main text for the completely symmetrized Pauli representation are given by:

$$F^{\boldsymbol{i},\lambda}_{q_\lambda,q'_\lambda} = \sum_{\substack{f_{11},f_{xx},f_{yy},f_{zz},\\ g_{010},g_{111},g_{0x1},g_{1x0},\\ g_{0y1},g_{1y0},g_{0z0},g_{1z1}\\ /Eq.\ equation\ C.22}} \frac{1}{\sqrt{\binom{n-2\lambda_1}{q_\lambda}\binom{n-2\lambda_1}{q'_\lambda}}} i^{2f_{xx}+2f_{yy}+2f_{zz}+2g_{1z1}-g_{0y1}+g_{1y0}} \cdot \frac{\lambda_1!(n-2\lambda_1)!}{f_{11}!f_{xx}!f_{yy}!f_{zz}!g_{010}!g_{111}!g_{0x1}!g_{1x0}!g_{0y1}!g_{1y0}!g_{0z0}!g_{1z1}!}\,, \tag{C.21}$$

where the sum is over a set of 12 non-negative integers fulfilling the constraints

$$\begin{aligned} g_{010}+g_{0z0}+g_{0x1}+g_{0y1} &= n-2\lambda_1-q_\lambda,\\ g_{010}+g_{0z0}+g_{1x0}+g_{1y0} &= n-2\lambda_1-q'_\lambda,\\ g_{111}+g_{1z1}+g_{1x0}+g_{1y0} &= q_\lambda,\\ g_{111}+g_{1z1}+g_{0x1}+g_{0y1} &= q'_\lambda,\\ 2f_{11}+g_{010}+g_{111} &= i_1,\\ 2f_{xx}+g_{0x1}+g_{1x0} &= i_x,\\ 2f_{yy}+g_{0y1}+g_{1y0} &= i_y,\\ 2f_{zz}+g_{0z0}+g_{1z1} &= i_z \end{aligned} \tag{C.22}$$

and λ_1 is the length of the second row of λ.

Proof. Following the previous section, we can project onto the space with an S_n irrep λ and a fixed multiplicity label $p_{\lambda 0}$ using the Young symmetrizer in Eq. equation C.8. Acting with the Young symmetrizer on a computational basis state yields a superposition of basis states with the same number of 0s and 1s. Let us write $\lambda = (\lambda_0, \lambda_1)$ for the lengths of the first and second row of λ. Then we see that applying the Young symmetrizer yields 0 unless the number of 1s is between λ_1 and λ_0. This is because the row symmetrizer does not change the number of 1s, and the antisymmetrizer on λ_1 length-2 columns yields 0 if any columns are 00 or 11. Thus, the irrep basis states can be obtained by applying the Young symmetrizer to states with $\lambda_1 + q_\lambda$ ones, where $0 \le q_\lambda \le n - 2\lambda_1$. Specifically, we can use

$$|\lambda, p_{\lambda 0}, q_\lambda\rangle = \Pi_{\lambda:p_{\lambda 0}} |x_{q_\lambda}\rangle\,, \tag{C.23}$$

with

$$|x_{q_\lambda}\rangle \coloneqq |01\rangle^{\otimes\lambda_1} \otimes |0\rangle^{\otimes n-2\lambda_1-q_\lambda} \otimes |1\rangle^{\otimes q_\lambda} \ . \tag{C.24}$$

Let us first evaluate

$$\sum_{r\in \mathrm{Row}(p_{\lambda 0})} R(r)\,|x_{q_\lambda}\rangle = \Pi_{r\to c}\left(|\Sigma^{q_\lambda}_{\lambda_0}\rangle \otimes |1\rangle^{\otimes\lambda_1}\right), \tag{C.25}$$

where $|\Sigma^y_x\rangle$ denotes the equal-weight superposition of all computation basis states on x qubits with $x-y$ zeros and y ones, which (up to normalization) is also known as *Dicke state* on x qubits [116, 36]. $\Pi_{r\to c}$ denotes the permutation of qubits needed to obtain the "column-standard" Young tableau $p_{\lambda 0}$ from an analogous "row-standard" Young tableau where the numbers first increase in the row direction and then in column direction. In other words, if we think of the qubits being associated to the tiles of the Young diagram λ, then the qubits in the first row are in state $|\Sigma^{q_\lambda}_{\lambda_0}\rangle$, and the qubits in the second row are in state $|1\rangle^{\otimes\lambda_1}$.

Next, for a two-row standard Young tableau $p_{\lambda 0}$, we have

$$\sum_{c\in \mathrm{Col}(p_{\lambda 0})} \mathrm{sgn}(c)R(c) = (\mathrm{id}_2 - \tau)^{\otimes\lambda_1} \otimes \mathrm{id}_2^{\otimes n-2\lambda_1} = (|\Psi\rangle\langle\Psi|)^{\otimes\lambda_1} \otimes \mathrm{id}_2^{\otimes n-2\lambda_1} \ , \tag{C.26}$$

where $|\Psi\rangle$ is the 2-qubit singlet state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \ , \tag{C.27}$$

and τ denotes the swap operator acting on two qubits. The qubits in the second row of λ in the state of Eq. equation C.25 are fixed to $|1\rangle$, so applying $|\Psi\rangle\langle\Psi|$ to each of the first λ_1 columns has the same effect as applying $|\Psi\rangle\langle\Psi|\,(|0\rangle\langle 0|\otimes \mathrm{id}_2)$. Applying $|0\rangle\langle 0|$ to the first λ_1 qubits of $|\Sigma^{q_\lambda}_{\lambda_0}\rangle$ yields $|0\rangle^{\otimes\lambda_1} \otimes |\Sigma^{q_\lambda}_{\lambda_0-\lambda_1}\rangle$. Thus, we find:

$$|\lambda, p_{\lambda 0}, q_\lambda\rangle = \Pi_{\lambda:p_{\lambda 0}}\,|x_{q_\lambda}\rangle = |\Psi\rangle^{\otimes\lambda_1} \otimes |\Sigma^{q_\lambda}_{n-2\lambda_1}\rangle\,. \tag{C.28}$$

Now, we are ready to evaluate

$$
\begin{aligned}
F^{\boldsymbol{i},\lambda}_{q_\lambda,q'_\lambda} &\coloneqq \langle \lambda, q_\lambda, p_{\lambda 0}| \, A_{\boldsymbol{i}} \, |\lambda, q'_\lambda, p_{\lambda 0}\rangle \\
&= (\langle\Psi|^{\otimes\lambda_1} \otimes \langle \Sigma^{q_\lambda}_{n-2\lambda_1}|) \left(\sum_{p_{\boldsymbol{i}}\in P_{\boldsymbol{i}}} p_{\boldsymbol{i}} \right) \left(|\Psi\rangle^{\otimes\lambda_1} \otimes |\Sigma^{q'_\lambda}_{n-2\lambda_1}\rangle \right) \\
&= \frac{1}{\sqrt{\binom{n-2\lambda_1}{q_\lambda}\binom{n-2\lambda_1}{q'_\lambda}}} \left(\sum_{s\in S^{q_\lambda}_{n-2\lambda_1}} \langle\Psi|^{\otimes\lambda_1} \otimes \langle s| \right) \left(\sum_{p_{\boldsymbol{i}}\in P_{\boldsymbol{i}}} p_{\boldsymbol{i}} \right) \left(\sum_{s'\in S^{q'_\lambda}_{n-2\lambda_1}} |\Psi\rangle^{\otimes\lambda_1} \otimes |s'\rangle \right) ,
\end{aligned}
\tag{C.29}
$$

where we used S^x_y to denote the set of bitstrings of length y with exactly x ones. This is a sum over (more than) exponentially many terms. Similarly to the previous Appendix, it can be evaluated efficiently by realizing that many summands have equal value. Thus, we instead sum over the different possible values multiplied with the number of summands with that value, which can be counted using combinatorics. Each summand is an overlap of two product states with a product operator in between. More precisely, we have a product of first λ_1 two-qubit overlaps, and then $n - 2\lambda_1$ single-qubit overlaps.

For each summand in Eq. equation C.29, let us denote by L_{ab} with $a, b \in \{1, x, y, z\}$ the subset of two-qubit pairs:

$$
L_{ab} \coloneqq \{(2l, 2l+1) : (p_{\boldsymbol{i}})_{2l} = \sigma_a, (p_{\boldsymbol{i}})_{2l+1} = \sigma_b, 0 \leq l < \lambda_1\} , \tag{C.30}
$$

and let us write $f_{ab} = |L_{ab}|$ for the number of elements in those subsets. The according overlap

$$
\langle\Psi| \, (\sigma_a \otimes \sigma_b) \, |\Psi\rangle \tag{C.31}
$$

is 0 if $a \neq b$, so we only need to consider subsets where $a = b$. The number of summands for given numbers f_{aa} is the number of decompositions of the first λ_1 qubit pairs into the four subsets L_{aa} with $a \in \{1, x, y, z\}$, which equals

$$
\frac{\lambda_1!}{f_{11}! f_{xx}! f_{yy}! f_{zz}!} . \tag{C.32}
$$

The value which the overlap on the first λ_1 qubit pairs contributes to each summand only depends on the numbers f_{aa}. The overlap in Eq. equation C.31 is given by 1 if $a = b = 1$,

and -1 if $a = b$ otherwise. Thus, the overall contribution to each summand is

$$(-1)^{f_{xx}+f_{yy}+f_{zz}} . \tag{C.33}$$

Next, let us consider the $n - 2\lambda_1$ single-qubit overlaps. For each summand in Eq. equation C.29, let us denote by K_{iaj} for $i, j \in \{0, 1\}$ and $a \in \{1, x, y, z\}$ the subset of the last $n - 2\lambda_1$ qubits

$$K_{iaj} \coloneqq \{l : (p_{\boldsymbol{i}})_{2\lambda_1+l} = \sigma_a, s_l = i, s'_l = j, 0 \leq l < n - 2\lambda_1\} , \tag{C.34}$$

and let us write $g_{iaj} = |K_{iaj}|$ for the number of elements in those subsets. The according overlap

$$\langle i| \sigma_a |j\rangle \tag{C.35}$$

is only non-zero if $i = j$ for $a \in \{1, z\}$ and $i \neq j$ for $a \in \{x, y\}$, so we can restrict to summands where only those 8 subsets are non-empty. The number of summands for given numbers g_{iaj} is the number of decompositions of the set of the last $n - 2\lambda_1$ qubits into the 8 subsets K_{iaj}, and is thus given by

$$\frac{(n - 2\lambda_1)!}{g_{010}!g_{111}!g_{0x1}!g_{1x0}!g_{0y1}!g_{1y0}!g_{0z0}!g_{1z1}!} . \tag{C.36}$$

The contribution of the overlap on the last $n-2\lambda_1$ qubits to each summand only depends on the numbers g_{iaj}. The single-qubit overlap in Eq. equation C.35 evaluates to 1 for g_{010}, g_{111}, g_{0X1}, g_{1X0} and g_{0z0}, -1 for g_{1z1}, i for g_{0y1}, and $-i$ for g_{1y0}. Thus the overall contribution to each summand is

$$(-1)^{g_{1z1}}(-i)^{g_{0y1}}(i)^{g_{1y0}} . \tag{C.37}$$

Overall, the number of summands for given f_{aa} and g_{iaj} is the product of Eq. equation C.32 and Eq. equation C.36, and the value of each summand is given by the product of Eq. equation C.33 and Eq. equation C.37. Plugging this into Eq. equation C.29 yields Eq. equation C.21. The constraints in Eq. equation C.22 are explained as follows. The first four constraints are due to the fact that the number of zeros and ones in s and s' is determined by q_λ and q'_λ, respectively. The last four constraints correspond to the fact that the number

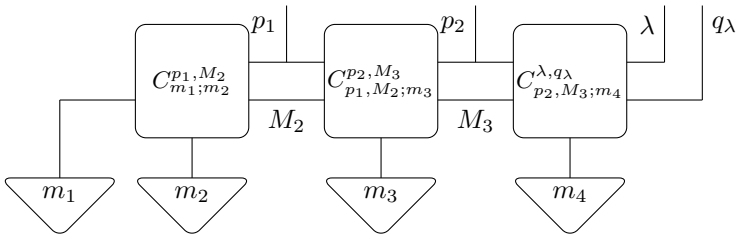


Figure C-2: A four-qubit example of the MPS giving $\langle m_1, m_2, m_3, m_4| \, |\lambda, q_\lambda, p_\lambda\rangle \, \langle \lambda, q_\lambda, p_\lambda|$.

of Pauli operators σ_1, σ_x, σ_y, σ_z in $p_{\boldsymbol{i}}$ is given by i_1, i_x, i_y, and i_z, respectively. □

In a similar fashion to the previous Appendix, we can easily evaluate the runtime this method achieves in calculating all of the matrix elements. Note that each component is a sum over four independent variables due to the constraints, yielding a runtime of $\mathrm{O}\left(n^4\right)$. Taking into account the $\mathrm{O}\left(n^6\right)$ tensor components of F yields the final runtime of $\mathrm{O}\left(n^{10}\right)$. Once again, it seems likely that the $\mathrm{O}\left(n^4\right)$ runtime for a single tensor component can be reduced to a smaller exponent. We will leave this open to further investigation.

C.4 End-to-End Classical Simulation From Tensor Networks

We here consider a slight variant of Corollary 8 where the inputs are given as classical matrix product state (MPS) descriptions rather than as quantum states. From [162], we have that $\langle m_1, m_2, m_3, m_4| \, |\lambda, q_\lambda, p_\lambda\rangle \, \langle \lambda, q_\lambda, p_\lambda|$ has an efficient MPS description, where $|m_1, m_2, m_3, m_4\rangle$ is a computational basis state; a four-qubit (i.e., $n = 4$) example is given in Fig. C-2, where we have used the Clebsch–Gordan coefficients $C^{\lambda,q_\lambda}_{p_1,m_1;p_2,m_2}$. p_i indices in Fig. C-2 are discarded for clarity where they are trivial. Note that these Clebsch–Gordan coefficients can be classically computed efficiently up to p bits of precision (i.e., up to additive error exponentially small in p) in time $\mathrm{poly}\left(n, p\right)$ by the Racah formula [281]. The indices associated with m_i can then be efficiently contracted with an efficient MPS description of an initial state—even if it is not permutation invariant on qubits—and the p_i indices efficiently traced out to efficiently yield matrix elements of $\tilde{\rho}$ as defined in Eq. (18) of the main text.

Appendix D

Appendix for Chapter 5

D.0.1 Projecting onto the group or algebra

Projections onto the orthogonal/unitary groups or their tangent spaces are central to optimizing over the space of orthogonal/unitary matrices. We focus the discussion here to the case of unitary matrices, but note that all of the following statements apply to orthogonal matrices as well by simple adjustments such as replacing the conjugate transpose (†) with the transpose ($^\intercal$). The tangent space $T_{\boldsymbol{U}}$ to a matrix $\boldsymbol{U} \in U(n)$ is equal to

$$T_{\boldsymbol{U}}U(n) = \left\{ \boldsymbol{X} \in \mathbb{C}^{n\times n} : \boldsymbol{U}^\dagger \boldsymbol{X} + \boldsymbol{X}^\dagger \boldsymbol{U} = 0 \right\}. \tag{D.1}$$

Given the canonical inner product $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \mathrm{Re}\left(\mathrm{Tr}[\boldsymbol{A}^\dagger \boldsymbol{B}]\right)$, we can show that the orthogonal projection onto the tangent space $T_{\boldsymbol{U}}U(n)$ is equal to that given by lemma 5.4.3 copied below.

Lemma 5.4.3 (Tangent space projection [336])**.** *Given the tangent space $T_{\boldsymbol{U}}U(n)$ of an orthogonal/unitary matrix $\boldsymbol{U}$, the orthogonal projection $\Pi_{T_{\boldsymbol{U}}}$ with respect to the canonical metric $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \mathrm{Re}\left(\mathrm{Tr}[\boldsymbol{X}^\dagger \boldsymbol{Y}]\right)$ is*

$$\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}) = \frac{1}{2}\left(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{X}^\dagger \boldsymbol{U}\right). \tag{5.5}$$

Similar to lemma 5.4.1, this projection also returns the closest matrix in Frobenius norm to $\boldsymbol{X}$ in the tangent space,

$$\min_{\boldsymbol{Y}\in T_{\boldsymbol{U}}U(n)} \|\boldsymbol{Y} - \boldsymbol{X}\|_F = \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}). \tag{5.6}$$

Proof. By the definition of an orthogonal projection, we need that $\Pi_{T_{\boldsymbol{U}}}(\Pi_{T_{\boldsymbol{U}}}(\cdot)) = \Pi_{T_{\boldsymbol{U}}}(\cdot)$ and for all $\boldsymbol{X}, \boldsymbol{Y} \in T_{\boldsymbol{U}}U(n)$ $\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}) = \boldsymbol{X}$ and $\langle \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}), \boldsymbol{X} - \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X})\rangle = 0$. The first two properties are straightforward to check. The last property can be shown as below using the definition of $\Pi_{T_{\boldsymbol{U}}}$ and cyclic property of trace:

$$\begin{aligned}
\langle \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}), \boldsymbol{X} - \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X})\rangle &= \frac{1}{4}\operatorname{Re}\Big(\operatorname{Tr}[(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U})^\dagger(\boldsymbol{X} + \boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U})]\Big) \\
&= \frac{1}{4}\operatorname{Re}\Big(\operatorname{Tr}[\boldsymbol{X}^\dagger\boldsymbol{X}] - \operatorname{Tr}[\boldsymbol{U}^\dagger\boldsymbol{X}\boldsymbol{X}^\dagger\boldsymbol{U}] + \operatorname{Tr}[\boldsymbol{X}^\dagger\boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U}] - \operatorname{Tr}[\boldsymbol{U}^\dagger\boldsymbol{X}\boldsymbol{U}^\dagger\boldsymbol{X}]\Big) \\
&= \frac{1}{4}\operatorname{Re}\Big(\operatorname{Tr}[(\boldsymbol{X}^\dagger\boldsymbol{U})^2] - \operatorname{Tr}[((\boldsymbol{X}^\dagger\boldsymbol{U})^2)^\dagger]\Big) \\
&= 0.
\end{aligned} \tag{D.2}$$

Furthermore, we have for all $\boldsymbol{Y} \in T_{\boldsymbol{U}}U(n)$ using triangle inequality and unitary invariance of the Frobenius norm:

$$\begin{aligned}
\|\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{X}) - \boldsymbol{X}\|_F &= \left\|\frac{1}{2}(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U}) - \boldsymbol{X}\right\|_F \\
&\leq \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\|_F + \frac{1}{2}\left\|\boldsymbol{Y} + \boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U}\right\|_F \\
&= \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\|_F + \frac{1}{2}\left\|-\boldsymbol{U}\boldsymbol{Y}^\dagger\boldsymbol{U} + \boldsymbol{U}\boldsymbol{X}^\dagger\boldsymbol{U}\right\|_F \\
&= \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\|_F + \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\|_F \\
&= \|\boldsymbol{Y} - \boldsymbol{X}\|_F,
\end{aligned} \tag{D.3}$$

which proves that $\Pi_{T_{\boldsymbol{U}}}(\cdot)$ projects onto the closest matrix $\boldsymbol{Y} \in T_{\boldsymbol{U}}U(n)$ in Frobenius norm. □

Since the set of unitary/orthogonal matrices does not form a vector space, an orthogonal projection is not a well defined operation in this space. However, it is still valid to ask what is the "closest" unitary/orthogonal matrix to a given matrix in a given norm. This is exactly what is stated in lemma 5.4.1 copied from the main text and proven below.

Lemma 5.4.1 (Projection onto unitary manifold [195])**.** *Given a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$:*

$$\Pi_U(\boldsymbol{A}) = \underset{\boldsymbol{U}\in\mathcal{U}(n)}{\arg\min}\, \|\boldsymbol{A} - \boldsymbol{U}\|_F^2 = \boldsymbol{A}(\boldsymbol{A}^\dagger\boldsymbol{A})^{-\frac{1}{2}}, \tag{5.3}$$

where $\mathcal{U}(n)$ indicates the set of $n \times n$ unitary matrices.

Proof. We follow the approach of [195] to prove this result. To shorten our notation, let $\boldsymbol{V} = \boldsymbol{A}(\boldsymbol{A}^\dagger \boldsymbol{A})^{-1/2}$. Given any unitary $\boldsymbol{U} \in U(n)$, let $\boldsymbol{U} = \boldsymbol{M} + \boldsymbol{V}$ for the properly chosen $\boldsymbol{M} \in \mathbb{C}^{n \times n}$. From unitarity of $\boldsymbol{U}$ and $\boldsymbol{V}$, we have

$$\boldsymbol{M}\boldsymbol{V}^\dagger + \boldsymbol{M}\boldsymbol{M}^\dagger + \boldsymbol{V}\boldsymbol{M}^\dagger = 0. \tag{D.4}$$

Then,

$$\begin{aligned} \|\boldsymbol{A} - \boldsymbol{U}\|_F^2 &= \|\boldsymbol{A} - \boldsymbol{M} - \boldsymbol{V}\|_F^2 \\ &= \|\boldsymbol{A} - \boldsymbol{V}\|_F^2 + \mathrm{Tr}[\boldsymbol{M}\boldsymbol{V}^\dagger + \boldsymbol{M}\boldsymbol{M}^\dagger + \boldsymbol{V}\boldsymbol{M}^\dagger] - \mathrm{Tr}[\boldsymbol{M}^\dagger \boldsymbol{A} + \boldsymbol{A}^\dagger \boldsymbol{M}] \\ &= \|\boldsymbol{A} - \boldsymbol{V}\|_F^2 - \mathrm{Tr}[\boldsymbol{M}^\dagger \boldsymbol{A} + \boldsymbol{A}^\dagger \boldsymbol{M}] \\ &= \|\boldsymbol{A} - \boldsymbol{V}\|_F^2 - \mathrm{Tr}[\boldsymbol{M}^\dagger \boldsymbol{V}(\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2} + (\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2}\boldsymbol{V}^\dagger \boldsymbol{M}], \end{aligned} \tag{D.5}$$

and since from eq. (D.4) we have that $\boldsymbol{M}\boldsymbol{V}^\dagger + \boldsymbol{V}\boldsymbol{M}^\dagger = -\boldsymbol{M}\boldsymbol{M}^\dagger$,

$$\|\boldsymbol{A} - \boldsymbol{U}\|_F^2 = \|\boldsymbol{A} - \boldsymbol{V}\|_F^2 + \mathrm{Tr}[(\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2}\boldsymbol{M}\boldsymbol{M}^\dagger]. \tag{D.6}$$

The second term above is non-negative since $\mathrm{Tr}[(\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2}\boldsymbol{M}\boldsymbol{M}^\dagger] = \mathrm{Tr}[\boldsymbol{M}^\dagger(\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2}\boldsymbol{M}]$ and $(\boldsymbol{A}^\dagger \boldsymbol{A})^{1/2}$ is positive semi-definite. Thus, for all $\boldsymbol{U} \in U(n)$,

$$\|\boldsymbol{A} - \boldsymbol{U}\|_F^2 \geq \|\boldsymbol{A} - \boldsymbol{V}\|_F^2, \tag{D.7}$$

which proves the result. □

D.1 Review of previous unitary neural network techniques

The integration of orthogonal/unitary matrices into neural networks is broadly aimed at maintaining stability in neural networks with many layers. For vanilla recurrent neural networks, repetitive application of the hidden-to-hidden transformation matrix exponentially amplifies or decays the eigenvalues of the transformation, thus resulting in exponentially large or small gradients. Similarly, in deep network architectures where weight matrices are drawn randomly, the norms of hidden states can similarly grow or decay exponentially with added layers. Enforcing unitarity or orthogonality of neural network layer transformations

offers a straightforward method to address these issues of instability since orthogonal/unitary matrices have eigenvalues of unity.

To establish notation and provide motivation for later analysis, consider a RNN whose input is a sequence of vectors $\boldsymbol{x}(t)$ with hidden layer $\boldsymbol{h}(t)$ updated according to the following rule:

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{M}\mathbf{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)}) \tag{D.8}$$

In the unitary or orthogonal formulation for RNNs, the matrix $\boldsymbol{W}$ in eq. (D.8) is replaced by a unitary or orthogonal matrix $\boldsymbol{U}$. During optimization of a unitary RNN, one must enforce the unitarity or orthogonality of $\boldsymbol{U}$ during training.

Existing methods to parameterize and enforce unitarity/orthogonality in neural network layers can be separated into three categories depending on the method of parameterization. We discuss each of these in detail below:

- **Layer-wise transformations:** among the first methods employed in this line of work, these methods parameterize orthogonal/unitary matrices in a layer-wise fashion where each layer is a parameterized orthogonal/unitary matrix. Example parameterizations include Givens rotations and Householder reflections. These methods are efficient when there are not many layers included in the parameterization and typically not employed when full access to the unitary/orthogonal group is required as full parameterization of an $n \times n$ unitary/orthogonal matrix requires $O(n)$ layers.

- **Lie algebra parameterization:** motivated by the fact that staying on the manifold of the Lie algebra is often easier than staying on the manifold of a Lie group, these methods parameterize the Lie algebra of the unitary/orthogonal groups and later obtain the actual unitary/orthogonal matrix by implementing the matrix exponential map. This matrix exponential map is typically the most costly step in these methods as one can either perform it directly (*e.g.,* using an SVD) or approximate it via Taylor series or Padé approximations which require repetitive application of matrices when applying it to an input. Though we adapt techniques from these methods in our work to efficiently perform gradient updates, we do not parameterize matrices in the Lie algebra.

- **Matrix entry parameterization:** as in typical neural network architectures, this method parameterizes an orthogonal/unitary matrix by directly parameterizing the entries of the matrix. This method is optimal in the "forward" direction since performing the transformation on an input simply requires matrix multiplication (as in vanilla architectures). However, updates to the matrix will no longer maintain unitarity or orthogonality, and one must employ methods to project these updates back onto the unitary/orthogonal manifold. This method is employed in our work.

We now present each of the above methods in the order given.

Layer-wise transformations Early algorithms [186, 24, 249] maintained unitarity by parameterizing a matrix $\boldsymbol{U}$ as a series or layered set of k parameterized unitary transformations:

$$\boldsymbol{U} = \boldsymbol{F}_{uni}^{(1)}(\theta_1)\boldsymbol{F}_{uni}^{(2)}(\theta_2)\cdots\boldsymbol{F}_{uni}^{(k)}(\theta_k). \tag{D.9}$$

where $\boldsymbol{F}_{uni}^{(i)}(\theta_i)$ indicates a transformation that maps parameters θ_i into a unitary matrix. These transformations include parameterized Givens rotations or Householder reflections. As a concrete example, consider the Givens rotation parameterization which is an orthogonal matrix that performs a rotation of the i, j-th dimensions by an amount θ:

$$\boldsymbol{G}(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, \tag{D.10}$$

or in other words, entries $G_{ii} = G_{jj} = \cos\theta$, $G_{ij} = -G_{ji} = \sin\theta$, and $G_{kl} = \delta_{kl}$ for all other entries.

In general, at least $k = O(n)$ layers are needed to achieve a full parameterization of the $n \times n$ unitary matrices. Training is performed by updating the parameters θ_i within each layer. However, for large matrices, due to the fact that $O(n)$ layers are required to parameterize the full space of transformations, these algorithms are only efficient when pa-

rameterizations over a subset of the space of unitary/orthogonal matrices suffices. Achieving this balance of parameterization versus performance is challenging as prior work – especially work studying the learnability of unitary matrices in quantum computation – has shown that loss landscapes over the unitary/orthogonal manifold may contain many bad local minima [206, 131, 14, 104].

Lie algebra parameterization As a reminder, the Lie algebra of the orthogonal and unitary groups are the set of skew symmetric ($\mathfrak{o}(n)$) and skew Hermitian ($\mathfrak{u}(n)$) matrices,

$$\mathfrak{o}(n) = \left\{ \boldsymbol{A} \in \mathbb{R}^{n \times n} : \boldsymbol{A} + \boldsymbol{A}^\intercal = 0 \right\}, \tag{D.11}$$

$$\mathfrak{u}(n) = \left\{ \boldsymbol{A} \in \mathbb{C}^{n \times n} : \boldsymbol{A} + \boldsymbol{A}^\dagger = 0 \right\}. \tag{D.12}$$

Transformations from the Lie algebra to the Lie group are performed using the exponential map which is surjective onto the connected components of the identity:

$$\exp(\boldsymbol{X}) = \sum_{k=0}^{\infty} \frac{\boldsymbol{X}^k}{k!} = \boldsymbol{I} + \boldsymbol{X} + \frac{1}{2}\boldsymbol{X}^2 + \frac{1}{6}\boldsymbol{X}^3 + \cdots \tag{D.13}$$

Thus, these methods parameterize the full space of unitary or special orthogonal matrices. However, the orthogonal group has two connected components (matrices with determinant equal to one and negative one) so the exponential maps only onto the positive determinant matrices.

Note that the Lie algebra is a vector space so the sum of two matrices in the Lie algebra is also in the algebra. Thus, gradient updates can typically be very easily performed. However, the exponential map is often expensive to compute, and much prior work has focused on implementing this transformation efficiently. [165] and [304] employ approximations to the matrix exponential via Padé or Taylor series approximations. For example, in the Taylor series approximation, one simply truncates the Taylor series to K entries:

$$\exp(\boldsymbol{X}) \approx \sum_{k=0}^{K} \frac{\boldsymbol{X}^k}{k!}, \qquad \left\| \exp(\boldsymbol{X}) - \sum_{k=0}^{K} \frac{\boldsymbol{X}^k}{k!} \right\|_2 \leq \frac{\|\boldsymbol{X}\|_2^k}{k!}, \tag{D.14}$$

where the norm $\|\cdot\|_2$ indicates the spectral norm (*i.e.,* largest singular value). This application of the matrix exponential comes at an added cost both when calculating derivatives and when applying the matrix in the forward direction (*e.g.,* when one simply desires the output

of a network). Applying the approximation above to the input of a layer requires $O(K)$ applications of the matrix. The matrix $\exp(\boldsymbol{X})$ can be explicitly constructed to avoid this added cost; however, obtaining this matrix requires a one time cost of $O(K)$ matrix-matrix multiplication operations which can be costly for large matrices.

The Padé approximants are more typically used in approximating the exponential map since they can guarantee that the approximated matrix is actually a unitary/orthogonal matrix (as opposed to a simple Taylor series approximation which does not provide this guarantee). A Padé approximant is an optimal rational function approximation to a given function. The Padé approximant to the exponential map takes the form $\exp(\boldsymbol{A}) \approx r_{mn}(\boldsymbol{A}) = p_m(\boldsymbol{A})q_n(\boldsymbol{A})^{-1}$ where $p_m(\cdot)$ and $q_n(\cdot)$ are order m and n polynomials respectively which have closed forms:

$$p_m(\boldsymbol{A}) = \sum_{k=0}^{m} \frac{(m+n-k)!m!}{(m+n)!(m-k)!k!}\boldsymbol{A}^k, \qquad q_n(\boldsymbol{A}) = \sum_{k=0}^{n} \frac{(m+n-k)!n!}{(m+n)!(n-k)!k!}(-\boldsymbol{A})^k. \quad \text{(D.15)}$$

The error of the approximation scales as $O(\|\boldsymbol{A}\|^{m+n+1})$ [168]. For unitary/orthogonal matrices, this approximation has the feature that for order $m = n$, applying the approximation to an element of the Lie algebra of the orthogonal/unitary matrices outputs a orthogonal/unitary matrix. Notably, setting $m = n = 1$ obtains the Cayley transform which was used in [165].

Finally, we note that approximations to the exponential function often can bias the gradients so in the RNN setting, [223] actually perform an SVD to obtain the actual output of the exponential map and analytically calculate gradients.

Matrix entry parameterization Parameterizing the entries of a matrix $\boldsymbol{U}$ directly is an obvious and simple means of constructing a unitary or orthogonal matrix. However, the set of unitary/orthogonal matrices do not form an algebra so one cannot simply update these matrices by simply adding a gradient update to a given matrix. Instead, updates to these matrices must be performed by projecting the updated matrix back onto the set of unitary/orthogonal matrices. Obviously, $\boldsymbol{U}$ must be initialized to be unitary/orthogonal in these methods.

Given a gradient update $\boldsymbol{G}$, the matrix $\boldsymbol{U} + \boldsymbol{G}$ is typically no longer unitary. Methods of Riemannian optimization are employed to update the matrix in a unitary/orthogonal

fashion. One means of updating the matrix is via the Cayley transform which computes a parametric curve in the direction, employed in [225, 336, 233, 315]. The Cayley transform "transports" $\boldsymbol{U}$ in the direction of the projection of the gradient in the tangent space $\Pi(\boldsymbol{G})$. *i.e.,* let $\boldsymbol{U}(0)$ be the initial unitary matrix, then one can transport the matrix in the direction of $\Pi(\boldsymbol{G})$ (where $\Pi(\cdot)$ is the projection onto the tangent space) by a "length" of α via the Cayley transform [261],

$$\boldsymbol{U}(\alpha) = \Big(\boldsymbol{I} - \frac{\alpha}{2}\Pi(\boldsymbol{G})\Big)^{-1}\Big(\boldsymbol{I} + \frac{\alpha}{2}\Pi(\boldsymbol{G})\Big)\boldsymbol{U}(0). \tag{D.16}$$

The above update formula requires a matrix inversion step which is the most costly step for large matrices. [225, 233] approximate the transformation via a fixed point interation which avoids having to invert a matrix but still requires matrix-matrix multiplication. More generally, the Cayley transform is a first order Padé approximant to the exponential map which provides a connection between the matrix entry parameterization and the Lie algebra parameterization discussed previously [168], *i.e.,* setting $m = n = 1$ in eq. (D.15) obtains the Cayley transform.

Our algorithms described in the main text directly parameterize matrix entries and thus follow this parameterization method. In performing the exponential map, we do not resort to any approximations since the exponential is efficient to perform in low rank settings. Updates to unitary matrices are either projected directly onto the closest unitary matrix in Frobenius norm (projUNN-D) or transported along the geodesic in the direction of the projection of the gradient onto the tangent space (projUNN-T). We refer the reader to the main text for a full description of our methodology.

D.1.1 Orthogonal or unitary convolution

Since convolutions are linear operators, one can perform orthogonal or unitary convolutions using similar techniques as in the general case studied above. As a reminder, for 2-D convolution, given input tensor $\mathbf{X} \in \mathbb{C}^{M\times N\times C}$ where C denotes the number of channels of the $M \times N$ input, linear convolution (or technically cross-correlation) with a filter $\mathbf{W} \in \mathbb{C}^{M\times N\times C\times C}$ takes the form

$$[\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{p,q,d} = [\mathbf{W} * \mathbf{X}]_{p,q,d} = \sum_{c=1}^{C}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbf{W}_{m,n,d,c}\mathbf{X}_{p+m,q+n,c}, \tag{D.17}$$

where the indexing above is assumed to be cyclic (taken modulus the corresponding dimension). Unitary or orthogonal convolutions form a subset of filters $\mathbf{W}$ which preserve the norm, *i.e.,* $\|\operatorname{conv}_{\mathbf{W}}(\mathbf{X})\| = \|\mathbf{X}\|$. Equivalently, $\operatorname{conv}_{\mathbf{W}}(\cdot)$ is orthogonal/unitary if the Jacobian of the transformation is also orthogonal/unitary.

The first method for orthogonalizing convolutions in neural networks was proposed in [296]. Their algorithm showed how to calculate the singular values of a linear convolution operation and then performed a series of projections on a given convolutional filter to project it onto an operator norm ball. Their algorithm made use of the convolution theorem which shows that linear convolution is diagonalized by Fourier transformations. [227] also proposed a method to orthogonalize convolutions by performing an SVD of the linearization of a convolution and then bounding the singular values accordingly. This method outputs linear transformations that are close to a convolution operation but no longer necessarily maintaining the equivariance of a standard convolution. Also, their method is expensive for large matrices as it requires implenting an SVD.

More recent methods implement orthogonal convolutions via approximations to the exponential map [315, 304]. These methods first form convolution filters $\mathbf{W}$ whose Jacobians $\mathbf{J}(\mathbf{W})$ satisfy the skew symmetry property of the orthogonal group:

$$\mathbf{J}(\mathbf{W}) = -\mathbf{J}(\mathbf{W})^{\mathsf{T}} \;\equiv\; \mathbf{W} = -\operatorname{conv\text{-}transpose}(\mathbf{W}), \tag{D.18}$$

where conv-transpose is the equivalent transposition operation in the filter space defined as [304]

$$[\operatorname{conv\text{-}transpose}(\mathbf{L})]_{m,n,c,d} = \mathbf{L}^{*}_{M-1-m,N-1-n,d,c} \tag{D.19}$$

for a filter $\mathbf{L} \in \mathbb{R}^{M\times N\times D\times C}$.

A given filter $\mathbf{W}$ can be transformed into a skew symmetric convolution filter by simply applying $\mathbf{L} = \mathbf{W} - \operatorname{conv\text{-}transpose}(\mathbf{W})$. As in the general case, one can apply the exponential map to the convolution filter to perform orthogonal convolution.

$$\exp[\operatorname{conv}_{\mathbf{L}}](\mathbf{X}) = \mathbf{X} + \mathbf{L} * \mathbf{X} + \frac{1}{2}\mathbf{L} *^2 \mathbf{X} + \frac{1}{6}\mathbf{L} *^3 \mathbf{X} + \cdots, \tag{D.20}$$

where $*^p$ indicates the convolution is applied p times, *e.g.,* $\mathbf{L} *^2 \mathbf{X} = \mathbf{L} * \mathbf{L} * \mathbf{X}$. Since the above operation is computationally expensive [304] implement a k-th order Taylor approximation

to the exponential map:

$$\exp[\text{conv}_{\mathbf{L}}](\mathbf{X}) \approx \sum_{p=0}^{k} \frac{1}{p!} \mathbf{L} *^{p} \mathbf{X}. \tag{D.21}$$

Given 2-D inputs of dimension $N \times N \times C$, applying convolution with filters with support over W elements in each dimension has runtime $O(pC^2W^2N^2)$. The added factor p in the runtime is required both upon training and evaluation of the network. Though this number is held constant, this adds a multiplicative overhead to running the algorithm both when evaluating an input and when training the algorithm. [304] use the above method to implement orthogonal convolution in their skew orthogonal convolution (SOC) algorithm. Similar techniques have been used to perform invertible (not necessarily orthogonal or unitary) convolutions in [172].

There are two key considerations or drawbacks associated with using the Taylor expansion to perform unitary/orthogonal convolution as in [304]. First, approximations via the Taylor approximation necessarily include an error that can be accounted for by scaling the factor p. However, unlike the Padé approximants discussed earlier such as the Cayley approximant, the Taylor approximation to the exponential map does not return a unitary/orthogonal operator. Thus, the value p needed to bound the error from any unitary/orthogonal operator also scales logarithmically with the desired error; however, this factor can also grow with the size of the filter, the number of channels, size of the input to a convolution operation, and especially the depth of the network. Second, Taylor approximations to a function can significantly bias the gradient. As noted in [223], their expRNN algorithm avoided using the Taylor or Padé approximation in the implementation of the unitary/orthogonal operation for this reason. As an example of how approximations can fail to represent the gradient, they provide the set of functions $f_n(x) = \sin(2\pi nx)/n$ approximating $f = 0$, where $f_n \to f$ uniformly but the derivatives do not converge to zero. Especially when constructing very deep networks, such instabilities can potentially cause issues in training.

Convolution operations over filters with a large support can be performed more efficiently in the Fourier domain as explored in various prior work [244, 40]. Specific to orthogonal convolution, [315] use the 2-D fast Fourier transform to more efficiently perform orthogonal convolution. For single channel inputs and outputs, convolution in the Fourier regime corresponds to pointwise multiplication over Fourier bases. Given multi-channel inputs and

outputs, convolution in the Fourier regime corresponds to matrix multiplication over the blocks indexed by channels where each block corresponds to a specific Fourier basis, *i.e.,* for a filter $\mathbf{W}$ and input $\mathbf{X}$:

$$[\text{FFT}\,\text{conv}_{\mathbf{W}}(\mathbf{X})]_{\widehat{r},\widehat{s},:} = \widehat{\mathbf{W}}^{*}_{\widehat{r},\widehat{s},:,:}\ [\text{FFT}\,\mathbf{X}]_{\widehat{r},\widehat{s},:}\ , \tag{D.22}$$

where $\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}$ is the representation of the filter in the Fourier regime. If the filter and and input are flattened in their spatial dimensions, we can represent the operation above as block-diagonal multiplication. For example, one can visually input the Fourier multiplication on an 2-channel input with spatial dimension $[2, 2]$.

$$[\text{FFT}\,\text{conv}_{\mathbf{W}}(\mathbf{X})] = \begin{bmatrix} \widehat{\mathbf{W}}^{*}_{1,1,1,1} & \widehat{\mathbf{W}}^{*}_{1,1,1,2} & & & & & & & & \\ \widehat{\mathbf{W}}^{*}_{1,1,2,1} & \widehat{\mathbf{W}}^{*}_{1,1,2,2} & & & & & & & & \\ & & \widehat{\mathbf{W}}^{*}_{1,2,1,1} & \widehat{\mathbf{W}}^{*}_{1,2,1,2} & & & & & & \\ & & \widehat{\mathbf{W}}^{*}_{1,2,2,1} & \widehat{\mathbf{W}}^{*}_{1,2,2,2} & & & & & & \\ & & & & \widehat{\mathbf{W}}^{*}_{2,1,1,1} & \widehat{\mathbf{W}}^{*}_{2,1,1,2} & & & & \\ & & & & \widehat{\mathbf{W}}^{*}_{2,1,2,1} & \widehat{\mathbf{W}}^{*}_{2,1,2,2} & & & & \\ & & & & & & \widehat{\mathbf{W}}^{*}_{2,2,1,1} & \widehat{\mathbf{W}}^{*}_{2,2,1,2} & & \\ & & & & & & \widehat{\mathbf{W}}^{*}_{2,2,2,1} & \widehat{\mathbf{W}}^{*}_{2,2,2,2} & & \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{X}}_{1,1,1} \\ \widehat{\mathbf{X}}_{1,1,2} \\ \widehat{\mathbf{X}}_{1,2,1} \\ \widehat{\mathbf{X}}_{1,2,2} \\ \widehat{\mathbf{X}}_{2,1,1} \\ \widehat{\mathbf{X}}_{2,1,2} \\ \widehat{\mathbf{X}}_{2,2,1} \\ \widehat{\mathbf{X}}_{2,2,2} \end{bmatrix}$$

[315] use the above technique, parameterizing filters in the Lie algebra of the orthogonal group. Filters are mapped into the orthogonal group using the Cayley transform (see eq. (D.16) for its form). Since the Cayley transform requires matrix inversion over the matrices above, their total runtime scales as $O(N^2C^2\log(N) + N^2C^3)$ operations for convolution over $N \times N$ images with C channels. Our methodology is adapted from the techniques in [315] and scales more efficiently as $O(N^2C\log(N) + kN^2C^2)$ time when the rank of gradient updates $k \ll N$. We instead parameterize the convolution filter directly in the Fourier regime, by parameterizing the individual blocks ($\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}$) of the block diagonal matrix above. One drawback of our method compared to [315] is that we cannot specify the support of the convolution in the real space of the Lie algebra as we parameterize matrices directly in the Fourier space. Of course, one can perform projections of the convolutions in Fourier space onto the bases spanned by the support of the elements in the Lie algebra; however, we opt instead to implement a regularizer which biases the filter towards elements of the desired space (*e.g.,* local elements).

Finally, [226] performed orthogonal convolution by parameterizing a block of matrices and corresponding projectors in their Block Convolutional Orthogonal Parametrization (BCOP) algorithm. Their method preceded those of [315] and [304]. However, their method

has two key drawbacks: it only parameterizes a subset of the space of orthogonal convolutions and is slower than other methods as it adds additional parameters to connect the various components in the space of orthogonal convolution.

Finally, methods have been proposed to iteratively or approximately maintain orthogonality. [35] propose a regularizer for convolutional layers which penalize matrices having singular values far from unity. [179] propose using a Newton's method iteration to update linear transformations to be closer to orthogonal/unitary. Their method requires iterative matrix-matrix multiplication over the unrolled weight matrix which can become expensive for large images.

D.1.2 Other related works

In the context of recurrent neural networks, designing RNNs to learn long sequences of data has a rich history of study. Some of the first and most celebrated algorithms include the long short-term memory networks (LSTM) [170] and gated reccurent unit (GRU) networks [91]. Since these works, various techniques have been used to more optimally avoid issues with learning long-sequence data. Beyond the unitary RNNs discussed earlier, some work has explored using bi-directional RNNs [160, 214], including those that have a more biologically inspired design [46]. These networks perform well on the copy task.

Another line of research studies the stability properties of continuous state space models which can be converted into a RNN formulation [125]. Some algorithms construct continuous state space models whose attractors are stable points in the dynamical system [81, 126]. More recently, continuous state space models have been designed with hidden state transformations that are customized to memorize data by limiting the learning over time to a subset of orthogonal polynomials [151, 152, 323]. Here, hidden states are in a sense parameterized over a set of polynomial coefficients [323]. These algorithms perform very well on tasks such as the copy task or Permuted MNIST but do not include unitary/orthogonal transformations in their network. In fact, more recent models [152, 323] achieve slightly higher scores on the TIMIT and permuted MNIST benchmarks compared to the unitary RNN formulations studied here.

In the convolutional setting, [339] form very deep convolutional neural networks by initializing parameters to construct norm-preserving orthogonal transformations. Orthogonality is not preserved during training however. [328] bias filters towards orthogonality by

implementing a regularizer that penalizes the weights when norms of outputs are larger or smaller than norms of inputs. The networks used in [339, 328] do not necessarily preserve the orthogonality property of their convolutional transformations during training. [192] study orthogonal convolutions from the basis of unitary/orthogonal wavelets showing how to represent convolution in terms of these wavelets.

Low rank approximations have been used in prior work in deep learning to prune neural network models [346, 311] and accelerate convolutions [184, 313, 183]. More related to our work, recent research has compressed gradients efficiently using low rank compression methods [324]. Although their focus was in sharing compressed information across computing units, their work lends support to the notion that the information of a gradient can be effectively and efficiently stored in low rank components. Research in learning theory has also noted connections between the stable rank of neural network parameters and generalization. [27] prove a generalization bound by compressing the models in the hypothesis class based on the stable rank of individual layers. [243] study the phases of learning from a random matrix theory setting showing that the stable rank of a matrix tends to decay over training. Various works have studied the implicit bias induced by optimization algorithms such as gradient descent showing that in many cases, the implicit bias is towards low rank solutions [156, 103]. Such bias towards low rank solutions has been explicitly proven in the setting of 2-layer matrix factorization [229], deep matrix factorization [25], linear group convolutional networks [219], and matrix recovery from Pauli measurements [234]. We note that relating low-rankness to the generalization ability of learning algorithms is a richly studied topic, and there are numerous papers that we did not mention here.

Finally, a wide range of work in quantum computation and quantum machine learning studies unitary learning algorithms in the context of quantum systems [50]. In fact, since the state space of closed quantum systems is transformed by unitary operators, quantum computers offer a unique platform for performing machine learning on the unitary manifold. This is an active area of research and existing methods for performing quantum machine learning on quantum architectures include variational algorithms which parameterize a quantum circuit and update the parameters via classical optimization methods [75, 334, 347, 198, 204, 327], quantum neural networks which design analogues to classical deep neural networks [209, 38, 295], and direct implementations of classical deep learning algorithms on quantum architectures [196, 72, 9]. We stress that these quantum algorithms are

inherently different in nature than their classical counterparts and there still exist significant challenges that must be surmounted before they become practically feasible [75, 50, 246]. For example, the loss landscapes of quantum algorithms often have many more poor local minima in comparison to classical counterparts [206, 14] and training of quantum architectures requires sampling of outputs which is challenging when derivatives decay with the size of a model – a phenomenon described as "barren plateaus" [246, 78]. Furthermore, even if algorithms can be efficiently run on a quantum computer, preparing data for use in a quantum computer and reading out information from the quantum computer are challenging tasks which are not guaranteed to be efficient [2].

D.2 Deferred proofs

D.2.1 Proof of Theorem 5.4.2

Recall Theorem 5.4.2:

Theorem 5.4.2 (Low rank unitary projection)**.** *Let $\boldsymbol{U}$ be an $n \times n$ orthogonal/unitary matrix perturbed by $\boldsymbol{G}_k$, a rank k matrix. Then the projection onto the closest orthogonal/unitary matrix defined below can be performed in $O(k(n^2 + nk + k^2))$ steps.*

$$\boldsymbol{U} + \boldsymbol{G}_k \to \underset{\boldsymbol{V} \in \mathcal{U}}{\arg\min} \|\boldsymbol{U} + \boldsymbol{G}_k - \boldsymbol{V}\|_F^2. \tag{5.4}$$

Proof. We proceed to prove the above statement by first analyzing the case where $k = 1$ and then generalizing to higher rank k. Recall from Lemma 5.4.1 that we would like to perform the following update:

$$\boldsymbol{U} + \boldsymbol{G}_k \to \underset{\boldsymbol{V} \in \mathcal{U}}{\arg\min} \|(\boldsymbol{U} + \boldsymbol{G}_k) - \boldsymbol{V}\|_F^2 = (\boldsymbol{U} + \boldsymbol{G}_k) \left[(\boldsymbol{U} + \boldsymbol{G}_k)^\dagger (\boldsymbol{U} + \boldsymbol{G}_k) \right]^{-\frac{1}{2}}. \tag{D.23}$$

Let the rank one vector components of $\boldsymbol{G}_k = \boldsymbol{a}\boldsymbol{b}^\dagger$ and define

$$\tilde{\boldsymbol{M}} = \boldsymbol{U} + \boldsymbol{G}_k = \boldsymbol{U} + \boldsymbol{a}\boldsymbol{b}^\dagger. \tag{D.24}$$

With the above, we can rewrite $(\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}})^{-\frac{1}{2}}$ in eq. (D.23) as:

$$\begin{aligned}(\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}})^{-\frac{1}{2}} &= [(\boldsymbol{U} + \boldsymbol{a}\boldsymbol{b}^\dagger)^\dagger (\boldsymbol{U} + \boldsymbol{a}\boldsymbol{b}^\dagger)]^{-\frac{1}{2}} \\ &= [\boldsymbol{I} + \hat{\boldsymbol{a}}\boldsymbol{b}^\dagger + \boldsymbol{b}\hat{\boldsymbol{a}}^\dagger + c_a \boldsymbol{b}\boldsymbol{b}^\dagger]^{-\frac{1}{2}},\end{aligned} \tag{D.25}$$

where $\hat{\boldsymbol{a}} = \tilde{\boldsymbol{U}}^\dagger \boldsymbol{a}$ and $c_a = \boldsymbol{a}^\dagger \boldsymbol{a}$.

eq. (D.25) is the Identity matrix plus the update of a rank two matrix. To see this, we decompose $\hat{\boldsymbol{a}}$ and $\boldsymbol{b}$ into orthogonal components using Gram Schmidt:

$$\boldsymbol{v}_1 = \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} \quad \boldsymbol{v}_2 = \frac{\hat{\boldsymbol{a}} - (\boldsymbol{v}_1^\dagger \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}}{\|\hat{\boldsymbol{a}} - (\boldsymbol{v}_1^\dagger \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}\|}. \tag{D.26}$$

In this new basis:

$$\hat{\boldsymbol{a}} = a_1 \boldsymbol{v}_1 + a_2 \boldsymbol{v}_2 \quad \boldsymbol{b} = b_1 \boldsymbol{v}_1, \tag{D.27}$$

and

$$\begin{aligned}&\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}} = \\ &\boldsymbol{I} + \begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2\end{bmatrix} \begin{bmatrix} a_1 b_1^* + b_1 a_1^* + c_a b_1 b_1^* & b_1 a_2^* \\ a_2 b_1^* & 0 \end{bmatrix} \begin{bmatrix}\boldsymbol{v}_1^\dagger \\ \boldsymbol{v}_2^\dagger\end{bmatrix}.\end{aligned} \tag{D.28}$$

Performing an eigendecomposition of the above 2×2 matrix into a diagonal eigenvalue matrix $\boldsymbol{S}$ and eigenvector matrix $\boldsymbol{C}$, we can rewrite $\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}}$ in a convenient form:

$$\begin{aligned}\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}} &= \boldsymbol{I} + \begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2\end{bmatrix} \boldsymbol{C} \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \boldsymbol{C}^\dagger \begin{bmatrix}\boldsymbol{v}_1^\dagger \\ \boldsymbol{v}_2^\dagger\end{bmatrix} \\ &= \boldsymbol{I} + s_1 \boldsymbol{u}_1 \boldsymbol{u}_1^\dagger + s_2 \boldsymbol{u}_2 \boldsymbol{u}_2^\dagger.\end{aligned} \tag{D.29}$$

Taking the inverse square root of the above can be performed by manipulating singular values:

$$\begin{aligned}(\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}})^{-\frac{1}{2}} = \boldsymbol{I} &+ ((s_1 + 1)^{-\frac{1}{2}} - 1)\boldsymbol{u}_1 \boldsymbol{u}_1^\dagger \\ &+ ((s_2 + 1)^{-\frac{1}{2}} - 1)\boldsymbol{u}_2 \boldsymbol{u}_2^\dagger.\end{aligned} \tag{D.30}$$

Finally, we multiply the above on the left by $\tilde{\boldsymbol{M}}$:

$$\tilde{\boldsymbol{M}}(\tilde{\boldsymbol{M}}^\dagger\tilde{\boldsymbol{M}})^{-\frac{1}{2}} = \tilde{\boldsymbol{M}} + ((s_1+1)^{-\frac{1}{2}} - 1)\tilde{\boldsymbol{M}}\boldsymbol{u}_1\boldsymbol{u}_1^\dagger + ((s_2+1)^{-\frac{1}{2}} - 1)\tilde{\boldsymbol{M}}\boldsymbol{u}_2\boldsymbol{u}_2^\dagger. \tag{D.31}$$

The above requires performing an eigendecomposition of a 2×2 matrix and a series of matrix-vector multiplication, matrix additions, and vector-vector outer products – in total scaling as $O(n^2)$ time.

Rank k updates Note that the above method can be extended to low rank updates, running in $O(kn^2)$ time when the rank of the update $k \ll n$. Specifically, now our update is:

$$\tilde{\boldsymbol{M}} = \boldsymbol{U} + \sum_{i=1}^{k} \boldsymbol{a}_i\boldsymbol{b}_i^\dagger. \tag{D.32}$$

Following the same steps would ultimately require performing an eigendecomposition of a $2k \times 2k$ matrix which takes $O(k^3)$ time. Additionally, one must perform Gram-Schmidt decomposition on a set of k vectors of length n which takes $O(k^2n)$ time. Finally, a series of $O(k)$ matrix-vector multiplications and vector-vector outer products is performed resulting in a total runtime of $O(k(n^2 + nk + k^2))$ time. In cases where $k \ll n$, the time to perform the Gram-Schmidt decomposition and the eigendecomposition is negligible and an overall runtime of $O(kn^2)$ time is achieved. Even in cases where updates are not low rank, one can apply efficient sampling procedures (see section 5.4.3) to find low rank approximations to the update matrix and apply the methods above while maintaining runtimes.

□

D.2.2 Proof of Theorem 5.4.4

Recall Theorem 5.4.4:

Theorem 5.4.4 (Low rank tangent transport)**.** *Let $\boldsymbol{U}$ be an $n \times n$ orthogonal/unitary matrix perturbed by $\boldsymbol{G}_k$, a rank k matrix. Then projecting $\boldsymbol{G}_k$ onto the tangent space and performing a rotation in that direction as defined in eq.* (5.7) *can be performed in $O(k(n^2 + nk + k^2))$ steps.*

Proof. We would like to efficiently perform the update below:

$$\boldsymbol{U} \to \boldsymbol{U} \exp\left[-\eta \boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k)\right]. \tag{D.33}$$

As before, we proceed to prove the above statement by first analyzing the case where $k = 1$ and then generalizing to higher rank k.

Let the rank one vector components of $\boldsymbol{G}_k = \boldsymbol{a}\boldsymbol{b}^\dagger$. Then

$$\boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k) = \frac{1}{2}\boldsymbol{U}^\dagger\left(\boldsymbol{G}_k - \boldsymbol{U}\boldsymbol{G}_k^\dagger \boldsymbol{U}\right) = \frac{1}{2}\left(\boldsymbol{U}^\dagger \boldsymbol{a}\boldsymbol{b}^\dagger - \boldsymbol{b}\boldsymbol{a}^\dagger \boldsymbol{U}\right) = \frac{1}{2}\left(\hat{\boldsymbol{a}}\boldsymbol{b}^\dagger - \boldsymbol{b}\hat{\boldsymbol{a}}^\dagger\right), \tag{D.34}$$

where $\hat{\boldsymbol{a}} = \boldsymbol{U}^\dagger \boldsymbol{a}$. The above is a rank 2 matrix. As before, we now proceed to perform an eigendecomposition in the low rank subspace of the above matrix. Using Gram Schmidt, we have

$$\boldsymbol{v}_1 = \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} \quad \boldsymbol{v}_2 = \frac{\hat{\boldsymbol{a}} - (\boldsymbol{v}_1^\dagger \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}}{\|\hat{\boldsymbol{a}} - (\boldsymbol{v}_1^\dagger \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}\|}. \tag{D.35}$$

In this new basis:

$$\hat{\boldsymbol{a}} = a_1 \boldsymbol{v}_1 + a_2 \boldsymbol{v}_2 \quad \boldsymbol{b} = b_1 \boldsymbol{v}_1, \tag{D.36}$$

and

$$\boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k) = \frac{1}{2}\begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2\end{bmatrix}\begin{bmatrix} a_1 b_1^* - b_1 a_1^* & -b_1 a_2^* \\ a_2 b_1^* & 0 \end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^\dagger \\ \boldsymbol{v}_2^\dagger\end{bmatrix}. \tag{D.37}$$

Performing an eigendecomposition of the above 2×2 matrix into a diagonal eigenvalue matrix $\boldsymbol{S}$ and eigenvector matrix $\boldsymbol{C}$, we can rewrite $\tilde{\boldsymbol{M}}^\dagger \tilde{\boldsymbol{M}}$ in a convenient form:

$$\begin{aligned}\boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k) &= \begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2\end{bmatrix}\boldsymbol{C}\begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}\boldsymbol{C}^\dagger \begin{bmatrix}\boldsymbol{v}_1^\dagger \\ \boldsymbol{v}_2^\dagger\end{bmatrix} \\ &= s_1 \boldsymbol{u}_1 \boldsymbol{u}_1^\dagger + s_2 \boldsymbol{u}_2 \boldsymbol{u}_2^\dagger.\end{aligned} \tag{D.38}$$

We apply the exponential map scaled by η to obtain

$$\begin{aligned}\exp\left[-\eta \boldsymbol{U}^\dagger \Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k)\right] = \boldsymbol{I} &+ \left(\exp(-\eta s_1) - 1\right)\boldsymbol{u}_1 \boldsymbol{u}_1^\dagger \\ &+ \left(\exp(-\eta s_2) - 1\right)\boldsymbol{u}_2 \boldsymbol{u}_2^\dagger.\end{aligned} \tag{D.39}$$

Finally, we multiply the above on the left by $\boldsymbol{U}$ to obtain the final result

$$\boldsymbol{U}\exp\left[-\eta\boldsymbol{U}^\dagger\Pi_{T_{\boldsymbol{U}}}(\boldsymbol{G}_k)\right] = \boldsymbol{U} + (\exp(-\eta s_1) - 1)\,\boldsymbol{U}\boldsymbol{u}_1\boldsymbol{u}_1^\dagger + (\exp(-\eta s_2) - 1)\,\boldsymbol{U}\boldsymbol{u}_2\boldsymbol{u}_2^\dagger. \tag{D.40}$$

The above requires performing an eigendecomposition of a 2×2 matrix and a series of matrix-vector multiplication, matrix additions, and vector-vector outer products – in total scaling as $O(n^2)$ time.

Rank k updates Note that as before, the above method can be extended to low rank updates, running in $O(kn^2)$ time when the rank of the update $k \ll n$. Specifically, now our update is:

$$\boldsymbol{U} \to \boldsymbol{U}\exp\left[-\eta\boldsymbol{U}^\dagger\Pi_{T_{\boldsymbol{U}}}\left(\sum_{i=1}^{k}\boldsymbol{a}_i\boldsymbol{b}_i^\dagger\right)\right]. \tag{D.41}$$

Following the same steps, the projection onto the tangent space would be a rank $2k$ matrix. The steps that follow would ultimately require performing an eigendecomposition of a $2k \times 2k$ matrix which takes $O(k^3)$ time. Additionally, one must perform Gram-Schmidt decomposition on a set of k vectors of length n which takes $O(k^2 n)$ time. Finally, a series of $O(k)$ matrix-vector multiplications and vector-vector outer products is performed resulting in a total runtime of $O(k(n^2 + nk + k^2))$ time. In cases where $k \ll n$, the time to perform the Gram-Schmidt decomposition and the eigendecomposition is negligible and an overall runtime of $O(kn^2)$ time is achieved. Even in cases where updates are not low rank, one can apply efficient sampling procedures (see section 5.4.3) to find low rank approximations to the update matrix and apply the methods above while maintaining runtimes.

□

D.2.3 First order equivalence to projUNN-D

Though projUNN-D and projUNN-T perform different updates, we can show that up to first order, the updates are in fact equivalent. Furthermore, as one may expect, this first order update is equal to the projection of the gradient onto the tangent space (see Lemma 5.4.3). In the case of projUNN-D, this shows that the update step is a retraction

or first order approximation to the matrix exponential implemented in projUNN-T [54].

Proposition D.2.1 (First order equivalence)**.** *For an $n \times n$ unitary/orthogonal matrix $\boldsymbol{U}$ perturbed by $\Delta\boldsymbol{U}$, gradient updates applied by algorithms* projUNN-D *(eq.* (5.4)*) and* projUNN-T *(eq.* (5.7)*) are, up to first order, equal to $\boldsymbol{U} + \Pi_{T_{\boldsymbol{U}}}(\Delta\boldsymbol{U})$, i.e.,*

$$\boldsymbol{U} \to \boldsymbol{U} + \frac{1}{2}\left(\Delta\boldsymbol{U} - \boldsymbol{U}\Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right) + O(\Delta\boldsymbol{U}\Delta\boldsymbol{U}^\dagger).$$

Proof. We first show that the above formula holds for projUNN-D and then show the same for projUNN-T. Recall the update formula for projUNN-D:

$$\boldsymbol{U} + \Delta\boldsymbol{U} \to \underset{\boldsymbol{V}\in\mathcal{U}}{\arg\min} \|(\boldsymbol{U} + \Delta\boldsymbol{U}) - \boldsymbol{V}\|_F^2 = (\boldsymbol{U} + \Delta\boldsymbol{U})\left[(\boldsymbol{U} + \Delta\boldsymbol{U})^\dagger(\boldsymbol{U} + \Delta\boldsymbol{U})\right]^{-\frac{1}{2}}. \quad \text{(D.42)}$$

Expanding the above up to first order and applying the first order Taylor expansion of $(\cdot)^{-1/2}$, we have:

$$\begin{aligned}(\boldsymbol{U} + \Delta\boldsymbol{U})\left[(\boldsymbol{U} + \Delta\boldsymbol{U})^\dagger(\boldsymbol{U} + \Delta\boldsymbol{U})\right]^{-\frac{1}{2}} &= (\boldsymbol{U} + \Delta\boldsymbol{U})\left[\boldsymbol{I} + \boldsymbol{U}^\dagger\Delta\boldsymbol{U} + \Delta\boldsymbol{U}^\dagger\boldsymbol{U} + O(\Delta\boldsymbol{U}^\dagger\Delta\boldsymbol{U})\right]^{-\frac{1}{2}} \\ &= (\boldsymbol{U} + \Delta\boldsymbol{U})\left[\boldsymbol{I} - \frac{1}{2}\left(\boldsymbol{U}^\dagger\Delta\boldsymbol{U} + \Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right) + O(\Delta\boldsymbol{U}^\dagger\Delta\boldsymbol{U})\right] \\ &= \boldsymbol{U} + \frac{1}{2}\left(\Delta\boldsymbol{U} - \boldsymbol{U}\Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right) + O(\Delta\boldsymbol{U}^\dagger\Delta\boldsymbol{U}).\end{aligned} \quad \text{(D.43)}$$

Similarly, for projUNN-T, recall the update formula (ignoring $-\eta$ term for learning rate):

$$\boldsymbol{U} \to \boldsymbol{U}\exp\left[\boldsymbol{U}^\dagger\Pi_{T_{\boldsymbol{U}}}(\Delta\boldsymbol{U})\right]. \quad \text{(D.44)}$$

Expanding the above and applying the first order approximation of the matrix exponential,

$$\begin{aligned}\boldsymbol{U}\exp\left[\boldsymbol{U}^\dagger\Pi_{T_{\boldsymbol{U}}}(\Delta\boldsymbol{U})\right] &= \boldsymbol{U}\exp\left[-\eta\boldsymbol{U}^\dagger\frac{1}{2}\left(\Delta\boldsymbol{U} - \boldsymbol{U}\Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right)\right] \\ &= \boldsymbol{U}\left[\boldsymbol{I} + \frac{1}{2}\left(\Delta\boldsymbol{U} - \boldsymbol{U}\Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right) + O(\Delta\boldsymbol{U}^\dagger\Delta\boldsymbol{U})\right] \\ &= \boldsymbol{U} + \frac{1}{2}\left(\Delta\boldsymbol{U} - \boldsymbol{U}\Delta\boldsymbol{U}^\dagger\boldsymbol{U}\right) + O(\Delta\boldsymbol{U}^\dagger\Delta\boldsymbol{U}).\end{aligned} \quad \text{(D.45)}$$

□

D.2.4 Unitary convolutional manifold is connected

Before proceeding to prove that the space of unitary convolutions is connected, we first provide a version of the convolution theorem which shows that convolution in the Fourier domain corresponds to block multiplication over channels. This is a classic result also contained in various prior works [296, 315], and we provide a short proof here for completeness.

Lemma D.2.2. *Given convolution filter* $\mathbf{W} \in \mathbb{C}^{M\times N\times C\times C}$ *and input* $\mathbf{X} \in \mathbb{C}^{M\times N\times C}$*, recall the definition of cyclic convolution (or technically cross-correlation) of* $\mathbf{W}$ *and* $\mathbf{X}$*:*

$$[\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{p,q,d} = \sum_{c=1}^{C}\sum_{m=1}^{M}\sum_{n=1}^{N} \mathbf{W}_{m,n,d,c}\mathbf{X}_{p+m,q+n,c}. \tag{D.46}$$

Let FFT *be the two dimensional fast Fourier transform, then convolution in the Fourier domain corresponds to block-wise multiplication over channels:*

$$[\mathrm{FFT}\,\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{\widehat{r},\widehat{s},:} = \widehat{\mathbf{W}}^{*}_{\widehat{r},\widehat{s},:,:}\ [\mathrm{FFT}\,\mathbf{X}]_{\widehat{r},\widehat{s},:}\,. \tag{D.47}$$

Proof. Let the roots of unity be denoted as $\omega_M = e^{2\pi i/M}$ and $\omega_N = e^{2\pi i/N}$. Then,

$$\begin{aligned}
[\mathrm{FFT}\,\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{\widehat{r},\widehat{s},d} &= \sum_{u=1}^{M}\sum_{v=1}^{N}\omega_M^{u\widehat{r}}\omega_M^{v\widehat{s}}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbf{W}_{m,n,d,c}\mathbf{X}_{u+m,v+n,c}\\
&= \sum_{u=1}^{M}\sum_{v=1}^{N}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{c=1}^{C}\omega_M^{u\widehat{r}}\omega_M^{v\widehat{s}}\omega_M^{\widehat{r}m}\omega_M^{-\widehat{r}m}\omega_N^{\widehat{s}n}\omega_N^{-\widehat{s}n}\mathbf{W}_{m,n,d,c}\mathbf{X}_{u+m,v+n,c}\\
&= \sum_{c=1}^{C}\sum_{m=1}^{M}\sum_{n=1}^{N}\omega_M^{-\widehat{r}m}\omega_N^{-\widehat{s}n}\mathbf{W}_{m,n,d,c}\sum_{u=1}^{M}\sum_{v=1}^{N}\omega_M^{\widehat{r}(u+m)}\omega_M^{\widehat{s}(v+n)}\mathbf{X}_{u+m,v+n,c}\mathbf{X}_{u+m,v+n,c}\\
&= \sum_{c=1}^{C}\widehat{\mathbf{W}}^{*}_{\widehat{r},\widehat{s},d,c}\widehat{\mathbf{X}}_{\widehat{r},\widehat{s},c}.
\end{aligned} \tag{D.48}$$

□

As an aside, the complex conjugation of the filter above is due to the fact that convolution in neural networks corresponds to the more commonly used term cross-correlation in mathematics. If the convolution operation were redefined as what is more commonly known as con-

volution in mathematics, *i.e.,* define conv′ as $[\text{conv}'_{\mathbf{W}}(\mathbf{X})]_{p,q,d} = \sum_{c=1}^{C}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbf{W}_{m,n,d,c}\mathbf{X}_{p-m,q-n,c}$, then the complex conjugate would no longer appear on the filter term.

From here, we simply show that each block in the above is connected which allows us to prove that the unitary manifold is connected.

Theorem 5.4.5 (Unitary convolutional manifold is connected)**.** *The space of unitary convolutions with filters of full support has a single connected component.*

Proof. Since the fourier transform is unitary and invertible, we can represent every filter $\mathbf{W}$ in the fourier domain as $\widehat{\mathbf{W}} \in \mathbb{C}^{M\times N\times C\times C}$ and vice-versa. In the Fourier domain, given lemma D.2.2, we have that convolution corresponds to block-wise multiplication over channels. For unitary convolution, each block $\widehat{W}_{\widehat{r},\widehat{s},:,:}$ indexed by frequencies $\widehat{r}$ and $\widehat{s}$ must be a unitary matrix, so we now analyze the set of filters whose blocks are unitary matrices in the Fourier domain.

The space of unitary matrices $U(C)$ is connected [159]. Therefore for every block, there exists a connected path between any two unitary matrices $\widehat{W}^{(1)}_{\widehat{r},\widehat{s},:,:}, \widehat{W}^{(2)}_{\widehat{r},\widehat{s},:,:} \in U(C)$. The full space of unitary convolutions is parameterized by the MN times direct product of groups $U(C)$, *i.e.,* $U(C)^{(\times MN)}$. Since the direct product of finitely many connected spaces is also connected, then the space of unitary convolutions is connected. □

D.3 Analysis on various benchmarked tasks

D.3.1 Learning random unitary

In addition to the analysis shown in the main text, here we include a plot showing the value of the Frobenius error as a function of the runtime (fig. D-1). Using projUNN-T to peform learning via low rank approximations to the gradient significantly speed up learning. Optimization was performed with vanilla gradient descent over batch sizes of 16 with learning rates of 0.5 and 0.33 for projUNN-T and projUNN-D respectively. Since the learning rate was fixed across all k for each of these experiments, the norm of the update was smaller for lower values of k. Scaling up the learning rate based on the value of k could make runtimes even faster for lower values of k.

For sake of completeness, we also include plots (in both regular and logarithmic scaling axis) showing the learning trajectory of the various combinations of samplers and pro-

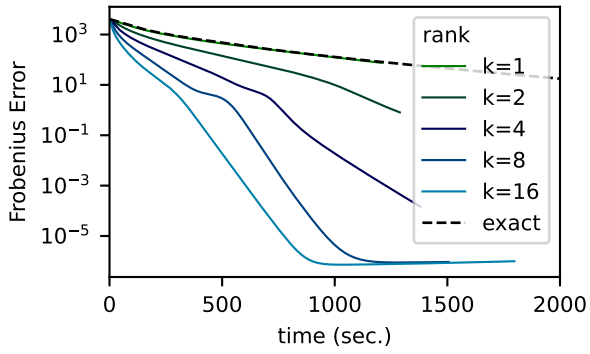


Figure D-1: Runtime of PROJUNN-T in learning a random target unitary matrix is faster when using low rank approximations. Here we plot Frobenius error $\|\boldsymbol{U} - \boldsymbol{U}_{tar}\|_F^2$ over the course of optimization. The learning rate is fixed for each value of k.

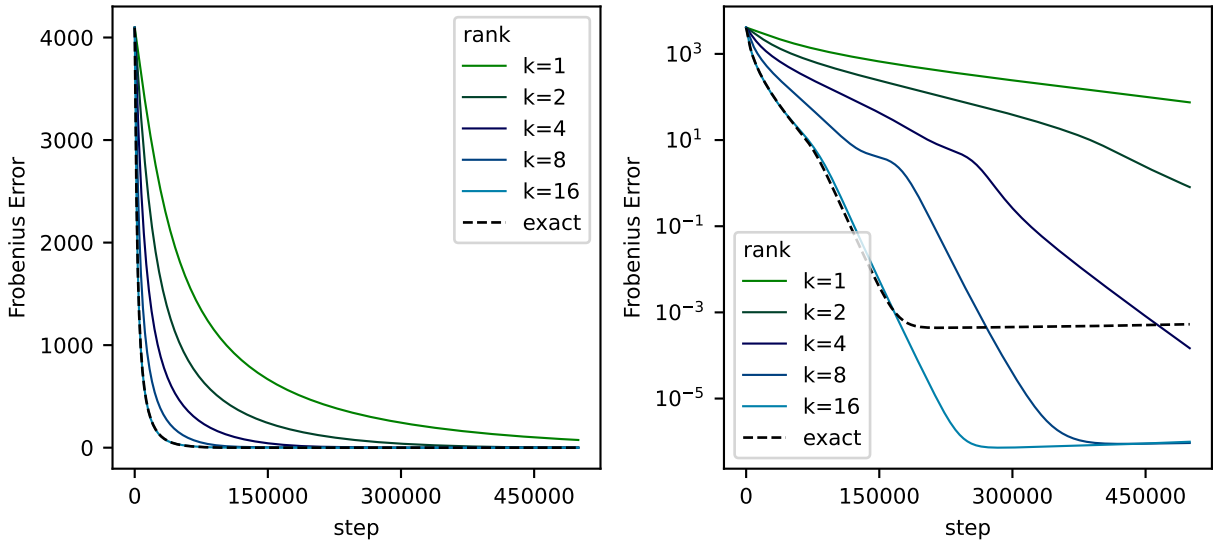


Figure D-2: Learning trajectory of PROJUNN-T equipped with the column sampling approximation in the random unitary learning task.

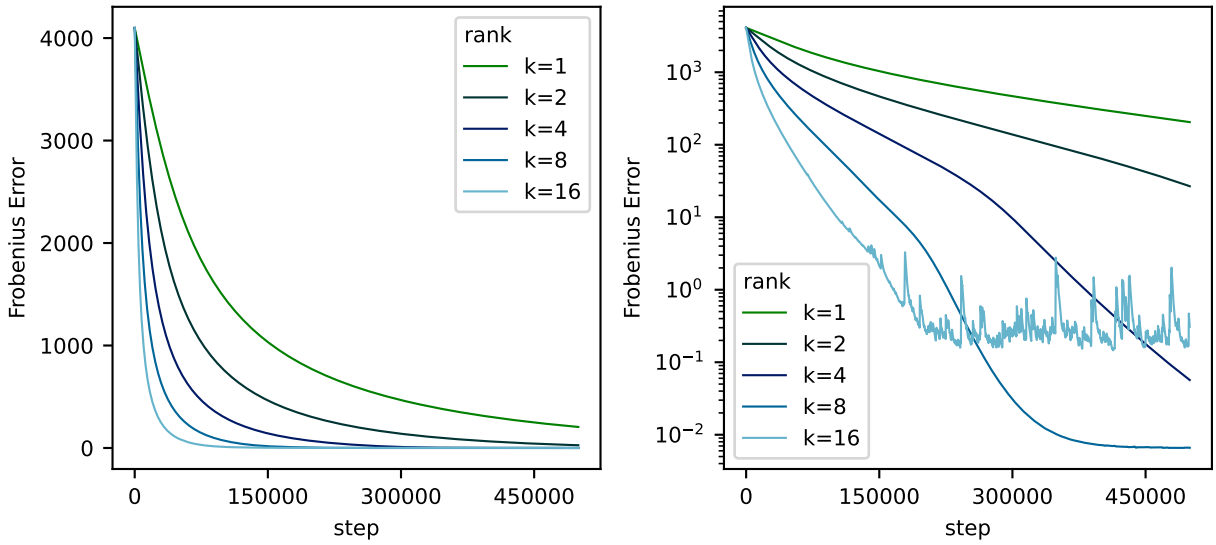


Figure D-3: Learning trajectory of projUNN-D equipped with the column sampling approximation in the random unitary learning task. Performing gradient updates using exact formulas was too computationally expensive for projUNN-D and thus not included here.

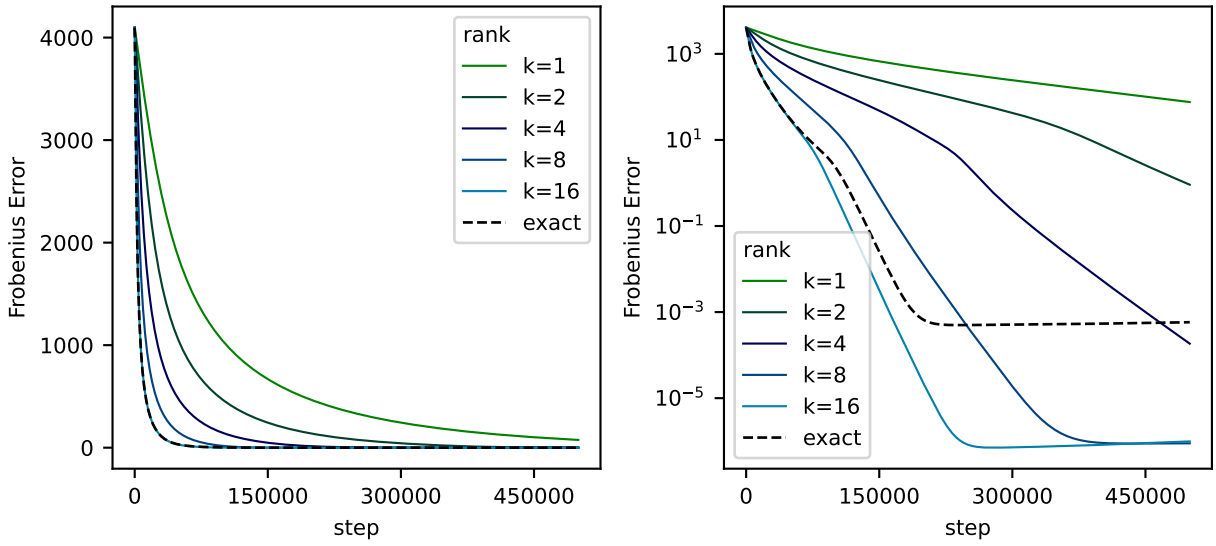


Figure D-4: Learning trajectory of projUNN-T equipped with the LSI sampling approximation in the random unitary learning task.

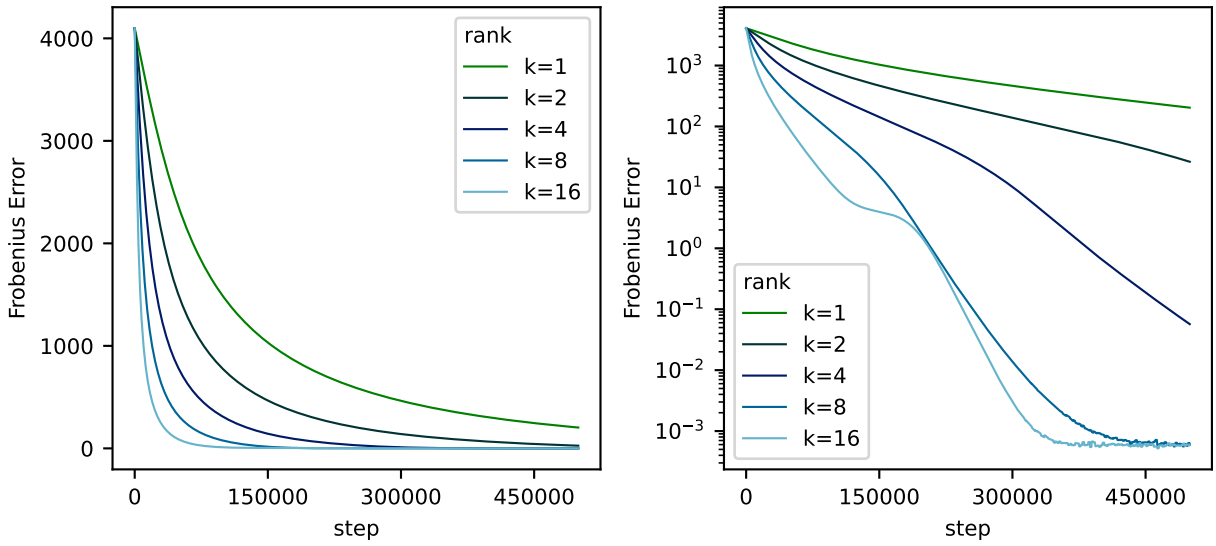


Figure D-5: Learning trajectory of projUNN-D equipped with the LSI sampling approximation in the random unitary learning task. Performing gradient updates using exact formulas was too computationally expensive for projUNN-D and thus not included here.

jUNN architectures in the random unitary task. See fig. D-2 for projUNN-T with column sampling (left hand side repeated from main text), fig. D-3 for projUNN-D with column sampling, fig. D-4 for projUNN-T with LSI sampling, and fig. D-5 for projUNN-D with LSI sampling.

fig. D-3 and fig. D-5 also highlight potential instabilities in training projUNN-D over long periods of time, a feature noted in the main text and discussed further in appendix D.6.3. To alleviate this, for projUNN-D architectures, we set the learning rate to slightly lower at 0.33 and projected parameter matrices onto the closest unitary using lemma 5.4.1 every 2048 steps. Note, that since this projection step was performed sparingly every n steps (where n is the dimension of the matrix), we did not find any significant decrease in runtime.

D.3.2 Adding task

As shown in the main text, our projUNN is able to learn the adding task even for long sequence lengths. Rank k gradient approximations were obtained using the column sampling approximation. RMSprop was used to train the RNN in this task. The learning rate was initialized to 0.001 for non-orthogonal parameters and 0.001/32 for all orthogonal parameters. Each epoch, the learning rate was reduced by multiplying it by 0.96. We found that initializing the orthogonal matrix as the identity matrix worked well for this task. This is

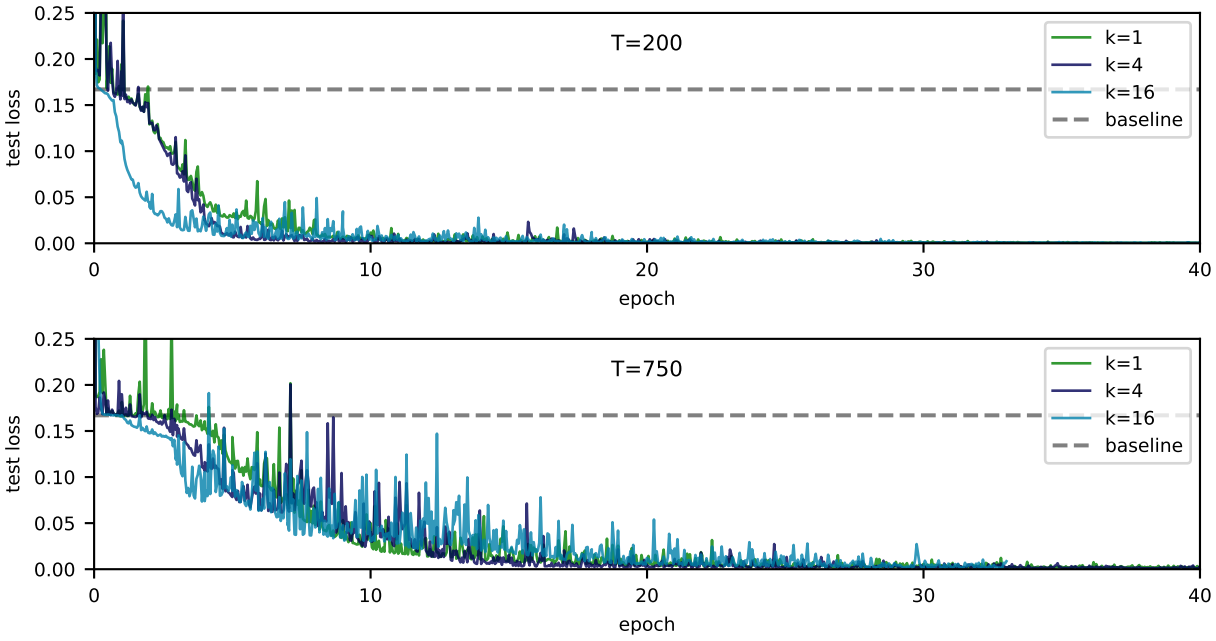


Figure D-6: projUNN-T (with column sampling approximation) is effective at learning the adding task. Above is a copy of fig. 5-3 except the plot here is expanded in size and test error is not smoothed.

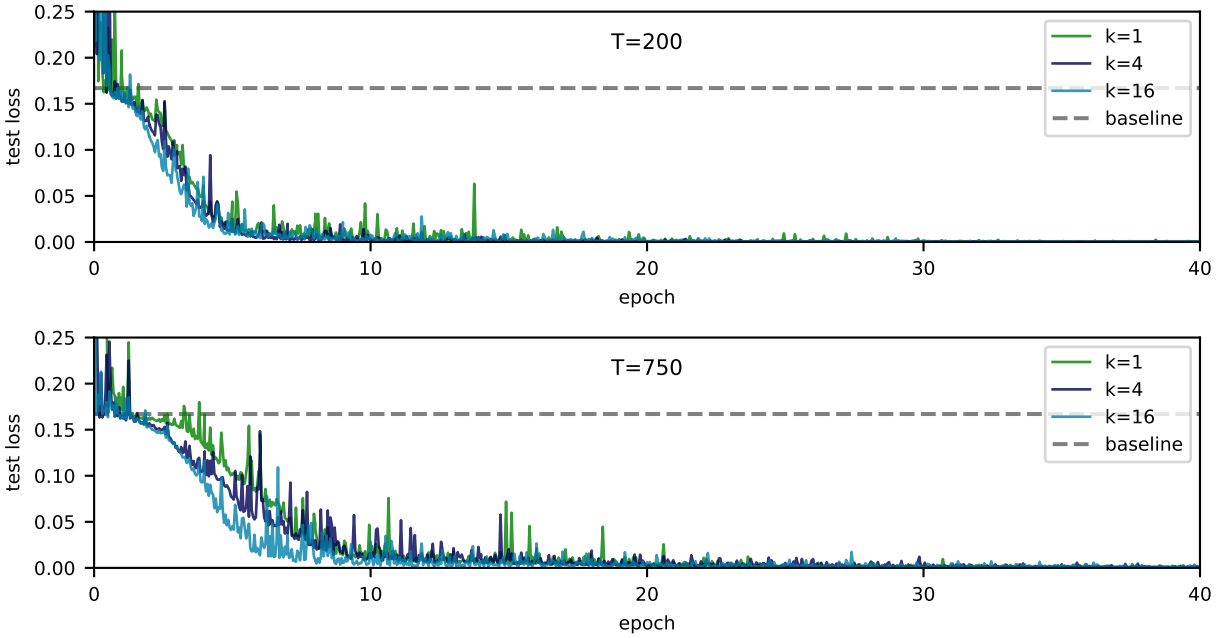


Figure D-7: projUNN-T (with LSI sampling approximation) is effective at learning the adding task.

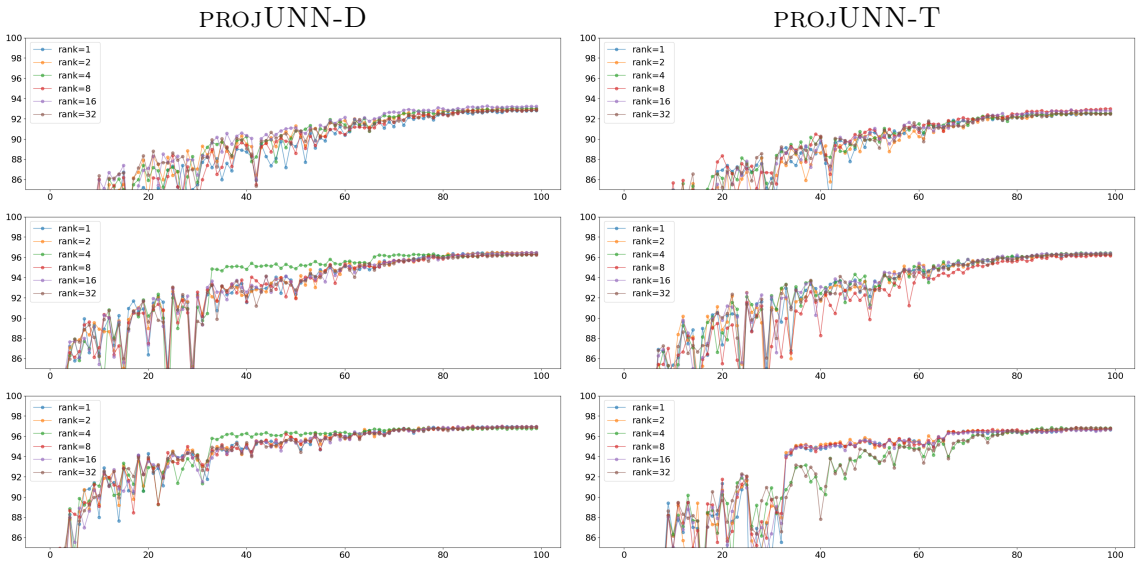


Figure D-8: Evolution of the test set accuracy during training at each epoch for the pixel-MNIST task. We depict the evolution for different RNN width (from top to bottom: 116,360 and 512). We observe that regardless of the rank k of the projUNN update, we reach the same final performances.

in distinction with *e.g.,* [165] which initialized using the Cayley initialization (block diagonal). For reference, we also include here an enhanced version of the plot in the main text in fig. D-6 and a similar plot obtained using projUNN with LSI sampling in fig. D-7.

D.3.3 Copy task

fig. 5-4 in the main text shows that our projUNN-T can efficiently learn the copy task. For this task, we set the learning rate of the RMSprop optimizer to $7e-4$. For orthogonal parameters, this learning rate was divided by 32. Orthogonal matrices were intialized using Henaff initialization [166] as described in appendix D.6.4. We found that this initialization scheme worked best in comparison to other methods.

D.3.4 Pixel permuted MNIST

fig. D-8 charts the trajectory of learning on the permuted MNIST task. As is evident in the figure, rank $k = 1$ is sufficient to guarantee optimal or nearly optimal convergence in all settings.

Table D.1: Test accuracy on CIFAR10 with different unitary convolution parameterizations and our proposed projUNN algorithm. We stress that SOC is an approximate unitary parameterization.

	Convolution Type					
Model	Standard	BCOP	SOC	Cayley	Proj-D	Proj-T
Resnet9	92.26	80.72	-	81.70	80.75	**82.06**
Resnet18	95.10	92.38	**94.24**	-	89.43	89.59

D.3.5 CIFAR10 CNN experiments

To explore the performance of our projUNN training algorithm for convolutional layers, we analyze its performance on CIFAR10 classification. Here, we provide further details to the results in the main text as well as some further preliminary experiments on unitary CNNs in resnet architectures. As in prior work [315] we employ the usual data-augmentation of random translations and left-right flips. We leverage the Resnet [163] architecture. As our previous analysis in the RNN setting has shown that rank $k = 1$ is sufficient for convergence, we always set $k = 1$ when using projUNN in the convolutional setting. We leverage the RMSprop optimizer and perform cross-validation on the learning rate. We present our results in table D.1 and make two observations. First, the smaller model (Resnet9) is able to reach or slightly outperform the alternative exact orthogonal constraints. Second, the larger model (Resnet18) falls behind the approximate orthogonal constraint method. This result is perhaps expected as our convolutional layers are full-width, *i.e.,* they allow for much greater degree of over-fitting. This was not detrimental in the small model with fewer convolutional layers. As a result, although we validate the ability of projUNN to produce state-of-the-art small convolutional networks, there remains open avenues of research to extend the method to larger models. As an aside, the resnet architecture is, in a sense, a very stable architecture since it is precisely designed to be able to incorporate many layers. To provide an honest comparison to prior work, we analyzed the performance of projUNN with respect to this resnet architecture, but note that more “vanilla" CNN architectures may be better targets for unitary constraints.

D.4 Analysis of low rank updates

It is typically the case that gradients of matrices with respect to a loss function are not *exactly* low rank but *approximately* low rank. More specifically, a matrix $\boldsymbol{A}$ is *approximately* low rank if there exists a rank k matrix $\boldsymbol{A}_k$ such that the relative error of the approximation E_{rel} is small:

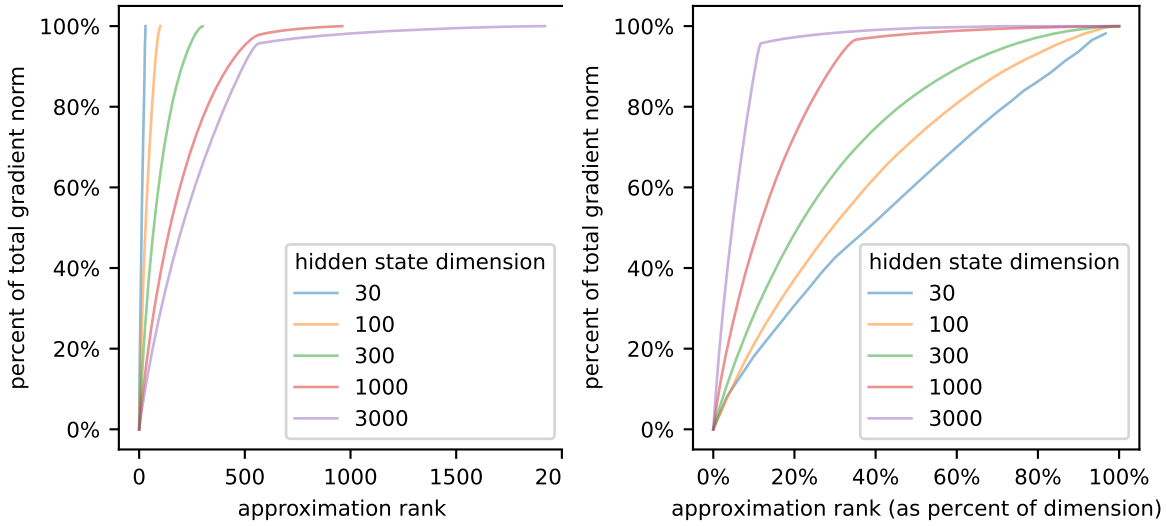
$$E_{rel} = \frac{\|\boldsymbol{A} - \boldsymbol{A}_k\|_F}{\|\boldsymbol{A}\|_F} \tag{D.49}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. We note that there is a connection between the above and the stable rank of a matrix $\boldsymbol{A}$ defined as $\|\boldsymbol{A}\|_F^2/\|\boldsymbol{A}\|_2^2$. The stable rank is upper bounded by the exact or hard rank of a matrix $\boldsymbol{A}$.

In RNN settings, fig. D-9 which plots $1 - E_{rel}$ shows that low rank approximations to a gradient can typically be very close to the true gradient even as the hidden size of the RNN is large. In fig. D-9, the RNN was given an input of batch size 64 where each input is a sequence of length 20 and each element of the sequence is a vector in $\mathbb{R}^{200}$ with elements drawn from the standard normal distribution. The RNN outputs logits to learn random target integers ranging from 1 to 10. Gradients were calculated for a single optimization step over a single batch of the random input and output data.

As observed in the main text, such low rank behavior is often more evident in real-world data. In fact, as seen in fig. 5-1a, virtually all the information of the gradient in a convolutional architecture is captured in the first few singular vectors. Gradients here are shown for a single $C \times C$ block in the Fourier regime of the convolution filter parameterized via our orthogonal PROJUNN convolution (see section 5.4.4 and appendix D.1.1 for form of parameterization). This filter is contained in the last residual block of the Resnet-9 network [163] and has 512 channels so the gradient is a 512×512 matrix. Since networks were trained with a batch size of 128, the maximum rank of this gradient is actually 128 (see short proof below in proposition D.4.2).

Since the weight matrix in hidden layers of RNNs is repetitively applied, gradients tend to be of a higher stable rank in these settings. Nevertheless, as evident in fig. D-10, gradients over the course of an epoch of training an RNN of hidden dimension 512 on the sequential MNIST task still exhibit clear low rank behavior, albeit not to the same degree as convolutional architectures. In constructing these plots, a batch size of 128 was used and networks were trained using the RMSprop algorithm. Throughout our experience with training the



(a) unnormalized

(b) normalized by dimension

Figure D-9: Portion of Frobenius norm (1-E_{rel}) captured by a low rank approximation to the gradient of the matrix typically improves with the dimension of the matrix. For larger matrices, at least half of the Frobenius norm of the gradient can be captured with a low rank approximation of a small percent of the overall dimension. Here, we assume that low rank approximations to the matrix are optimal and the RNN is trained on random inputs and outputs. Plots are for the matrix performing transformation from hidden states to hidden states (*i.e.,* the matrix replaced by a unitary/orthogonal matrix in PROJUNN implementations.

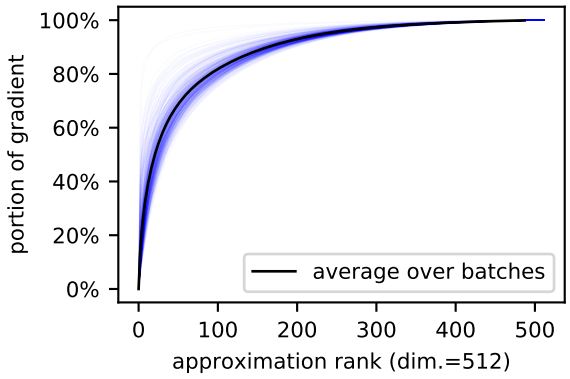


Figure D-10: Low rank approximations capture most of the Frobenius norm of the gradient. Here, we plot gradients of the matrix in the RNN's hidden layer for each batch (light blue line) over an epoch of training PROJUNN in the pixel-by-pixel MNIST task (see section 5.5).

PROJUNN on various tasks, we observe that setting the rank of an approximation to even just one is effective at ensuring convergence of training to a good solution.

We furthermore note that the exact rank of a gradient in neural network architectures can typically be bounded by the dimensions of the input. For convolutional networks, this exact rank is bounded by the batch size and for vanilla recurrent neural networks, this exact rank is bounded by the batch size times the sequence length. We provide formal statements and short proofs of these propositions below.

Proposition D.4.1. *Given a loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ taking in two real numbers and outputting a real number, let $f^{(t)}$ denote the t-th sequential output of a vanilla RNN defined as*

$$\begin{aligned} \boldsymbol{h}^{(t)}(\boldsymbol{x}) &= \sigma(\boldsymbol{M}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)}(\boldsymbol{x})) \\ f^{(t)}(\boldsymbol{x}) &= \tau(\boldsymbol{V}\boldsymbol{h}^{(t)}(\boldsymbol{x})), \end{aligned} \tag{D.50}$$

where $t \in [T]$ indexes the sequence of inputs, inputs $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$ ($\boldsymbol{x}$ denotes the concatenation of all inputs), input transformation $\boldsymbol{M} \in \mathbb{R}^{h \times d}$, hidden transformation $\boldsymbol{W} \in \mathbb{R}^{h \times h}$, output transformation $\boldsymbol{V} \in \mathbb{R}^{h \times d}$, and σ and τ are pointwise non-linearities. Let $\nabla_{\boldsymbol{W}} L$ indicate the gradient of the hidden transformation matrix $\boldsymbol{W}$ with respect to the loss function $L = \sum_{i=1}^{b} \ell(f^{(T)}(\boldsymbol{x}_i), y_i)$ over a batch of inputs $\{\boldsymbol{x}_i, y_i\}_{i=1}^{b}$. The rank of $\nabla_{\boldsymbol{W}} L$ is bounded above by bT.

Proof. Define $\boldsymbol{z}^{(t)} = \boldsymbol{M}\mathbf{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)}(\boldsymbol{x})$. Calculating the gradient of the loss function with respect to the parameter $\boldsymbol{W}$, we have:

$$\begin{aligned} \nabla_{\boldsymbol{W}} L &= \sum_{i=1}^{b} \nabla_{\boldsymbol{W}} \ell(f^{(T)}(\boldsymbol{x}_i), y_i) \\ &= \sum_{i=1}^{b} \sum_{t=1}^{T} \sum_{k=1}^{h} \left[\frac{\partial \ell(f^{(T)}(\boldsymbol{x}_i), y_i)}{\partial \boldsymbol{z}^{(t)}} \right]_k \frac{\partial \boldsymbol{z}_k^{(t)}}{\partial \boldsymbol{W}} \\ &= \sum_{i=1}^{b} \sum_{t=1}^{T} \left[\frac{\partial \ell(f^{(T)}(\boldsymbol{x}_i), y_i)}{\partial \boldsymbol{z}^{(t)}} \right] \left[\boldsymbol{h}^{(t-1)}(\boldsymbol{x}_i) \right]^{\intercal}. \end{aligned} \tag{D.51}$$

The above is the sum of bT rank one matrices which is at most rank bT concluding the proof. □

Proposition D.4.2. *Given a loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ taking in two real numbers and outputting a real number and convolution filter $\mathbf{W} \in \mathbb{R}^{N \times M \times C \times C}$, let $f(\mathbf{X})$ denote the convolutional neural network defined as*

$$\begin{aligned} \mathbf{H}(\mathbf{X}) &= g_{pre}(\mathbf{X}) \\ \mathbf{Y}(\mathbf{X}) &= \mathrm{conv}_{\mathbf{W}}(\mathbf{H}(\mathbf{X})) \\ f(\mathbf{X}) &= g_{post}(\mathbf{Y}(\mathbf{X})), \end{aligned} \tag{D.52}$$

where input $\mathbf{X} \in \mathbb{R}^{N_{in} \times M_{in} \times C_{in}}$ and g_{pre} and g_{post} are functions which apply the layers before and after the convolution with filter $\mathbf{W}$. Let $\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}$ denote the $C \times C$ matrix storing the values of the convolution filter in the Fourier regime for frequencies $\widehat{r}$ and $\widehat{s}$ (see eq. (D.22)). Let $\nabla_{\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}} L$ indicate the gradient of $\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}$ with respect to the loss function $L = \sum_{i=1}^{b} \ell(f(\mathbf{X}_i), y_i)$ over a batch of inputs $\{\mathbf{X}_i, y_i\}_{i=1}^{b}$. Then, for all $\widehat{r}$ and $\widehat{s}$ in the support of the filter, the rank of $\nabla_{\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}} L$ is bounded above by b.

Proof. We have that

$$\nabla_{\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}} L = \sum_{i=1}^{b} \nabla_{\widehat{\mathbf{W}}_{\widehat{r},\widehat{s},:,:}} \ell(f(\mathbf{X}_i), y_i). \tag{D.53}$$

We now proceed to show that the rank of $\nabla_{\widehat{\mathbf{W}}_{:,:,\widehat{r},\widehat{s}}} \ell(f(\mathbf{X}_i), y_i)$ is equal to one which completes the proof. As a reminder, we have via the convolution theorem that

$$[\mathrm{FFT}\,\mathrm{conv}_{\mathbf{W}}(\mathbf{X})]_{\widehat{r},\widehat{s},:} = \widehat{\mathbf{W}}^{*}_{\widehat{r},\widehat{s},:,:}\ [\mathrm{FFT}\,\mathbf{X}]_{\widehat{r},\widehat{s},:}\,. \tag{D.54}$$

Let $\widehat{\mathbf{Y}}(\mathbf{X}) = \mathrm{FFT}\,\mathrm{conv}_{\mathbf{W}}(\mathbf{X})$, then we have

$$\begin{aligned} \nabla_{\widehat{\mathbf{W}}_{:,:,\widehat{r},\widehat{s}}} \ell(f(\mathbf{X}_i), y_i) &= \sum_{k=1}^{C} \frac{\partial \ell(f(\mathbf{X}_i), y_i)}{\partial \widehat{\mathbf{Y}}(\mathbf{X})_{\widehat{r},\widehat{s},k}} \frac{\partial \widehat{\mathbf{Y}}(\mathbf{X})_{\widehat{r},\widehat{s},k}}{\mathbf{W}_{\widehat{r},\widehat{s},:,:}} \\ &= \frac{\partial \ell(f(\mathbf{X}_i), y_i)}{\partial \widehat{\mathbf{Y}}(\mathbf{X})_{\widehat{r},\widehat{s},:}} \left([\mathrm{FFT}\,\mathbf{X}]_{\widehat{r},\widehat{s},:} \right)^{\mathsf{T}}. \end{aligned} \tag{D.55}$$

The above is a rank one matrix which concludes the proof. □

D.4.1 Other sampling algorithms

The prior analysis showed that gradients are typically low rank, and in the main text, we listed a couple efficient algorithms that allow one to approximate a full but approximately low rank matrix with an explicitly low rank matrix. Enhancements to these algorithms exist that may provide further improvements in very large dimensions. For example, the *FKV sampling* [132] and *constant time SVD* [118] algorithms provide runtimes that are often logarithmic in the dimension of the matrix for sampling entries from the low rank approximation to a matrix. Nevertheless, explicitly constructing all entries of a rank k approximation to an $n \times n$ matrix $\boldsymbol{A}$ requires time at least $O(k^2 n)$. Improvements in runtime are achieved by applying random projections to both the column and row subspaces of the matrix $\boldsymbol{A}$ to perform the final approximation. In practice, the runtime of these algorithms depends on other factors that limit the applicability of the algorithm even for relatively large matrices. Numerical simulations show that FKV sampling achieves a practical speedup for matrices of dimension approximately 10^6 or higher [29]. Since matrices in our projUNN were of much smaller dimension, we did not use these methods.

D.5 Runtime comparisons

table 5.1 (see main text) and table D.2 lists the asymptotic runtime complexities of unitary/orthogonal models in the RNN and convolutional setting respectively. For RNNs, projUNN has the best runtime scaling of all models which parameterize the full orthogonal/unitary space with just one layer. In fact, apart from the rank approximation factor k, the runtime of projUNN is optimal in the RNN setting. In the convolutional setting, projUNN scales most efficiently when there are many channels or the filter size W scales faster than the logarithm of the input dimension $\log N$. For small filter sizes, BCOP and SOC scale relatively efficiently. In fact, SOC is nearly optimal for $W \ll \log N$ up to the approximation factor p in the Taylor expansion which is required both upon evaluation and training of the network.

Runtimes for RNN implementations in practice Enforcing unitarity in projUNN incurs a computational overhead associated to performing eigendecompositions and QR decompositions entailed in updating the gradient. Even though such computations are

Model	Complexity of forward + backward step	Notes
BCOP [1]	$O(C^2W^2N^2 + C^3W^3)$	W sequential orthogonalizations needed (slow for large W)
SOC [2]	$O(pC^2W^2N^2)^a$	p denotes number of terms in Taylor series approximation
Cayley [3]	$O(N^2C^2\log(N) + N^2C^3)^b$	
projUNN (our method)	$O(N^2C\log(N) + kN^2C^2)^c$	k denotes rank of gradient updates

[1] [226], [2] [304], [3] [315]

[a] just applying the convolution (without gradient update) also requires the added factor p in runtime unlike standard convolutions and BCOP which run in time $O(C^2W^2N^2)$ upon just evaluation, [b] approximations to matrix inversion exist which may reduce runtimes though these approximations are not implemented here, [c] runtime shown for typical setting when $k \ll n$

Table D.2: Time complexity of 2-D orthogonal convolutional layers with filter size $W \times W$ applied to $N \times N$ inputs with C input and output channels.

performed on a small subspace of the total dimension of the matrix, such computations may not effect increase training times by a constant factor as evident in fig. 5-2a. Empirically, we observed that much of the increased overhead was due to performing eigendecompositions and QR decompositions on a GPU, both tasks which are challenging to parallelize on GPU architectures. We note that similar issues may be one reason why the expRNN runtimes appear much slower in our simulations as shown in fig. 5-2a. Recent updates to pytorch [270] and tensorflow [3] have focused on improving the runtimes of these common linear algebraic routines, and we expect these runtimes to continue to improve in the future.

Runtimes for convolutional network implementations in practice fig. D-11, fig. D-12, and fig. D-13 compare runtimes of orthogonal convolution algorithms when varying the number of channels, input size, and filter size respectively. fig. D-14 plots the runtime when the filter and image size are set equal to each other and varied together. In summary, we find that the skew orthogonal convolution (SOC) [304], the Cayley model [315], and our projUNN are all relatively efficient for small to medium sized models. projUNN empirically performs best in comparison to other models when there are many channels or the filter size is large. SOC [304] is the fastest when the filter size is small. As a reminder, SOC [304] employs the Taylor series approximation to form a unitary/orthogonal matrix. We did not change the number of terms in the approximation used even though increasing the size of a filter, dimension of an input, or the number of channels will likely require more terms to maintain the same error in the approximation.

The Cayley model [315] has a runtime that is empirically similar to projUNN in most of our experiments despite projUNN having an improved asymptotic scaling with regards

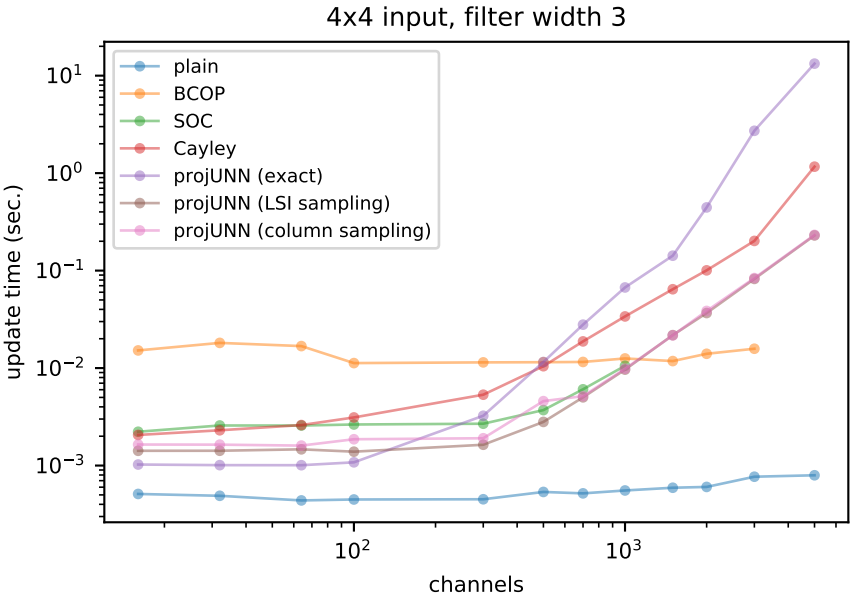


Figure D-11: Comparison of runtimes of orthogonal convolutional architectures when varying the number of channels (log-log plot). The number of input channels is equal to the number of output channels. Runtimes are averaged over 10 forward and backward passes through a single convolutional operation on random data. PROJUNN with LSI sampling and column sampling have very similar runtimes and may appear to completely overlap on the chart. Out of memory error obtained for BCOP for large number of channels.

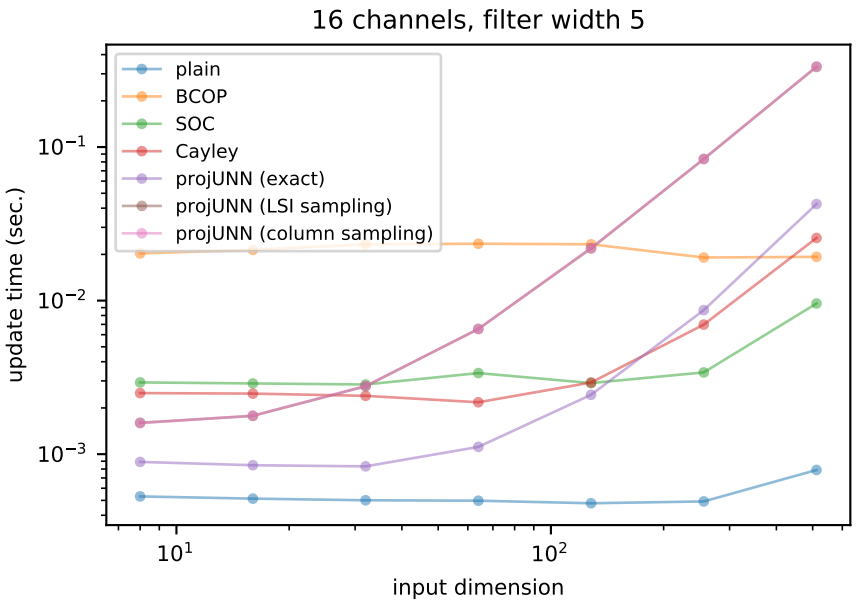


Figure D-12: Comparison of runtimes of orthogonal convolutional architectures when varying the size of the input which is a square image (log-log plot). Runtimes are averaged over 10 forward and backward passes through a single convolutional operation on random data. PROJUNN with LSI sampling and column sampling have very similar runtimes and may appear to completely overlap on the chart.

to the number of channels (see table D.2). Runtimes here are calculated by averaging across the first 10 steps of gradient descent on random data for initialized networks. Close to initialization, matrices in the convolution filter are generally well conditioned and this may be one reason why the Cayley model showed good scaling in fig. D-11 since the matrix inversion for the well conditioned matrices runs quickly. We also suspect that for the number of channels considered, the added costs of performing sampling and projections in PROJUNN do not materialize into a very clear runtime benefit. Interestingly, the exact version of PROJUNN (no low rank approximation) runs in similar time to the Cayley method.

Finally, we note that in comparison to RNN implementations, there is increased overhead associated to performing eigendecompositions and QR decompositions with our projUNN convolutional network implementations since unitary matrices are batched across the elements of the convolutional filter. As mentioned before, much of the increased overhead was due to performing eigendecompositions and QR decompositions on a GPU, both tasks which can be challenging to parallelize on GPU based architectures. Future updates to Pytorch [270] and Tensorflow [3] may help improve runtimes of these operations.

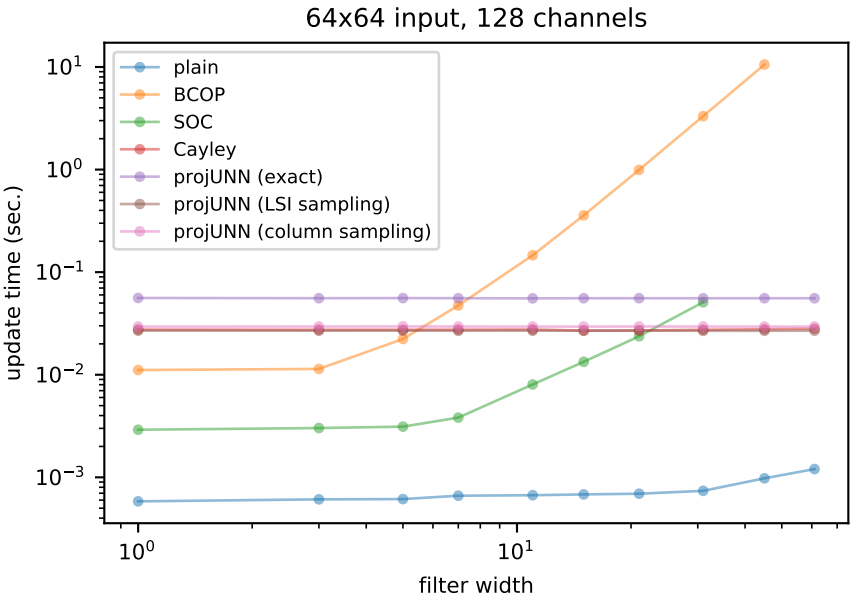


Figure D-13: Comparison of runtimes of orthogonal convolutional architectures when varying the size of the filter (log-log plot). Both PROJUNN and the cayley convolution [315] perform operations on the full space of convolution filters so the filter size does not change the runtime for these models. Runtimes are averaged over 10 forward and backward passes through a single convolutional operation on random data. PROJUNN with LSI sampling and column sampling have very similar runtimes and may appear to completely overlap on the chart. Out of memory error obtained for SOC beyond filter size of 31.

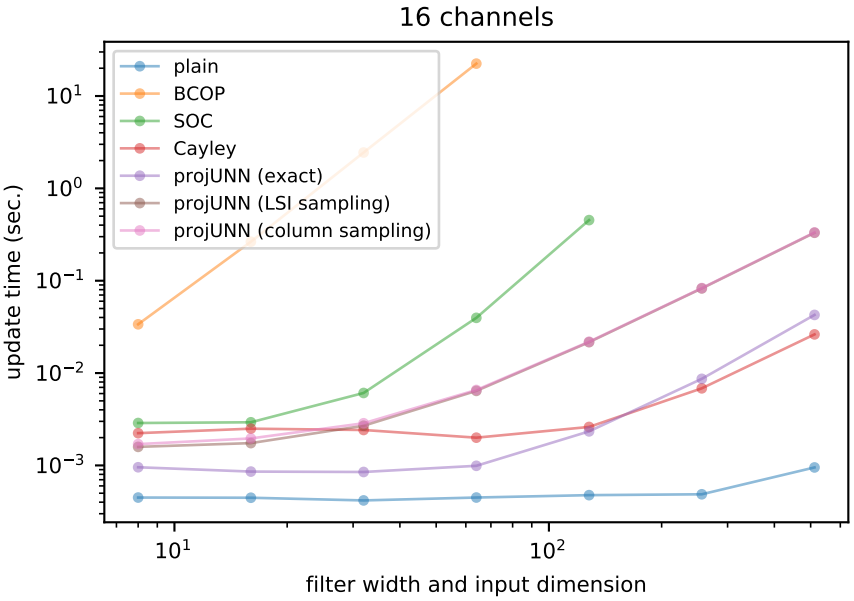


Figure D-14: Comparison of runtimes of orthogonal convolutional architectures when varying the size of the filter and the input dimension in-tandem (log-log plot). Runtimes are averaged over 10 forward and backward passes through a single convolutional operation on random data. PROJUNN with LSI sampling and column sampling have very similar runtimes and may appear to completely overlap on the chart. Out of memory error obtained for SOC beyond filter size and image size of 128.

D.6 Network architectures and training details

ExpRNN trivializations	https://github.com/Lezcano/expRNN	MIT
scoRNN	https://github.com/SpartinStuff/scoRNN	NA
scuRNN	https://github.com/Gayan225/scuRNN	NA
skewed orthogonal convolutions (SOC)	https://github.com/singlasahil14/SOC	NA
Cayley convolution	https://github.com/locuslab/orthogonal-convolutions	MIT
BCOP	github.com/ColinQiyangLi/LConvNet	MIT

D.6.1 Handling complex numbers

Since matrices in the unitary group are complex valued, care must be taken to ensure that the neural network can handle such complex numbers. In these settings, standard activation functions need to be adapted for complex numbers. It is often advantageous to have a nonlinearity which does not change the phase of its input. For these reasons, the activation function used is typically the modRELU activation shown below:

$$\begin{aligned} \sigma_{modRELU}(z) &= (|z| + b)\frac{z}{|z|} \text{ if } \|z\| + b > 0 \\ \sigma_{modRELU}(z) &= 0 \text{ if } |z| + b \leq 0 \end{aligned} \tag{D.56}$$

where b is a bias term (trainable). In calculations of the modulus $|z|$, a small additive constant is often added to avoid stability issues when $|z|$ is small.

Another challenge that arises with performing orthogonal convolution is that the representation of a real-valued convolutional filter in the Fourier domain will be complex-valued. More specifically, the fast Fourier transform of a real signal is Hermitian-symmetric. For example, for a one dimensional vector $\boldsymbol{x} \in \mathbb{R}^N$ where $\widehat{\boldsymbol{x}} = \text{FFT}\,\boldsymbol{x} \in \mathbb{C}^N$ and $\widehat{\boldsymbol{x}}_i = \widehat{\boldsymbol{x}}_{-i \mod N}$. Therefore, when initializing weight filters in our orthogonal convolution operations, one must be careful to ensure the Hermitian symmetric property holds. Similarly, when performing convolutions in the Fourier space, care must be taken in converting data types to and from complex and real space. In our implementation, we used the pytorch FFT.RFFT2 and FFT.IRFFT2 commands to implement these operations efficiently [269].

D.6.2 Optimizers

For training projUNN architectures, a separate optimizer is used for unitary/orthogonal parameters which must be projected after updates and all other parameters. Learning rates for the unitary/orthogonal parameters are typically set to less than that of the other parameters. In our implementations, we found that a learning rate for the unitary or orthogonal parameters of one-tenth or one-twentieth of that of the other parameters works well in practice. Furthermore, when using optimizers with momentum terms, we constructed a variant of standard optimizers to apply changes to the momentum terms after projecting gradients onto the unitary/orthogonal manifold or tangent space. This optimizer performed well in our experiments though sometimes added instability. Therefore, unless otherwise stated, standard optimizers were used.

D.6.3 Numerical stability

Though updates via projUNN mathematically guarantee that parameters remain unitary/orthogonal, performing a significant number of sequential updates to a matrix can add numerical errors over time slowly drifting parameters away from the unitary/orthogonal manifold. This is especially true in the case of projUNN-D where updating matrices requires division of eigenvalues in the eigendecompositon of the rank k subspace (see appendix D.2 and eq. (D.38)). In contrast, we empirically find that projUNN-T maintains stability throughout optimization even when requiring many updates. To actively address potential instabilities, one can perform eigendecompositions with higher floating point precision or sporadically project unitary/orthogonal parameters onto the closest unitary/orthogonal matrix via lemma 5.4.1. Though this projection step requires $O(n^3)$ time for an $n \times n$ matrix, performing this projection only every $O(n)$ gradient updates can still preserve efficient runtimes of on average $O(kn^2)$ time per gradient update. Unless otherwise stated, we do not perform any additional steps for maintaining stability.

D.6.4 Initialization of unitary/orthogonal parameters

Empirically, we found that initializing unitary/orthogonal matrices to be Haar random or close to Haar random does not perform well. This is also an observation noted in prior works [166, 165, 223]. Instead, we used one of two different initialization schemes. The

first initializes unitary/orthogonal parameters as the identity matrix. The second initializes unitary/orthogonal matrices using variants of the so-called Henaff initialization [166] where 2×2 diagonal blocks of the matrices are initialized as samples from the below

$$\exp\left(\begin{bmatrix} 0 & s_i \\ -s_i & 0 \end{bmatrix}\right), \tag{D.57}$$

where s_i is sampled uniformly from $[-\pi, \pi]$ [166]. In other words, an $n \times n$ matrix $\boldsymbol{U}$ is initialized as

$$\boldsymbol{U} = \begin{bmatrix} \exp\left(\begin{bmatrix} 0 & s_1 \\ -s_1 & 0 \end{bmatrix}\right) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \exp\left(\begin{bmatrix} 0 & s_2 \\ -s_2 & 0 \end{bmatrix}\right) & & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \exp\left(\begin{bmatrix} 0 & s_{n/2} \\ -s_{n/2} & 0 \end{bmatrix}\right) \end{bmatrix}. \tag{D.58}$$

Note, that since we parameterize matrices in the Lie group instead of the Lie algebra, we include the exponential map in the parameterization above. Other variants of the above have been used in prior works. For example, the Cayley initialization samples $s_i = \sqrt{\frac{1-\cos t_i}{1+\cos t_i}}$ where t_i is sampled uniformly from $[0, \pi/2]$ [165].

D.6.5 Computational details

We employed the latest version of PyTorch (1.10.1+cu102) with Python3.8 and all the pre-requisite libraries coming along. The hardware leverages Quadro GP100 GPU and Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz.

Bibliography

[1] Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3089–3114, 2007.

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

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] J Agredo. On exponential convergence of generic quantum Markov semigroups in a Wasserstein-type distance. *International Journal of Pure and Applied Mathematics*, 107(4):909–925, 2016.

[5] J Agredo and Franco Fagnola. On quantum versions of the classical Wasserstein distance. *Stochastics*, 89(6-7):910–922, 2017.

[6] Julián Agredo. A Wasserstein-type distance to measure deviation from equilibrium of quantum Markov semigroups. *Open Systems & Information Dynamics*, 20(02):1350009, 2013.

[7] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the Jones polynomial. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 427–436, New York, NY, USA, 2006. Association for Computing Machinery.

[8] Shahnawaz Ahmed, Carlos Sánchez Muñoz, Franco Nori, and Anton Frisk Kockum. Quantum state tomography with conditional generative adversarial networks. *arXiv preprint arXiv:2008.03240*, 2020.

[9] Jonathan Allcock, Chang-Yu Hsieh, Iordanis Kerenidis, and Shengyu Zhang. Quantum algorithms for feedforward neural networks. *ACM Transactions on Quantum Computing*, 1(1):1–24, 2020.

[10] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[11] Mohammad H. Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum Boltzmann machine. *Phys. Rev. X*, 8:021050, 5 2018.

[12] Abhinav Anand, Jonathan Romero, Matthias Degroote, and Alán Aspuru-Guzik. Experimental demonstration of a quantum generative adversarial network for continuous distributions. *arXiv preprint arXiv:2006.01976*, 2020.

[13] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 500–510, USA, 2014. Society for Industrial and Applied Mathematics.

[14] Eric R Anschuetz. Critical points in hamiltonian agnostic variational quantum algorithms. *arXiv preprint arXiv:2109.06957*, 2021.

[15] Eric R Anschuetz, Andreas Bauer, Bobak T Kiani, and Seth Lloyd. Efficient classical algorithms for simulating symmetric quantum systems. *arXiv preprint arXiv:2211.16998*, 2022.

[16] Eric R. Anschuetz and Yudong Cao. Realizing quantum Boltzmann machines through eigenstate thermalization, 2019.

[17] Eric R Anschuetz, David Gamarnik, and Bobak Kiani. Combinatorial nlts from the overlap gap property. *arXiv preprint arXiv:2304.00643*, 2023.

[18] Eric R Anschuetz and Bobak T Kiani. Quantum variational algorithms are swamped with traps. *Nature Communications*, 13(1):7760, 2022.

[19] Eric R. Anschuetz and Bobak T. Kiani. Quantum variational algorithms are swamped with traps. *Nat. Commun.*, 13:7760, 12 2022.

[20] Eric Ricardo Anschuetz. Critical points in quantum generative models. In *International Conference on Learning Representations*, 2022.

[21] Anurag Anshu, Srinivasan Arunachalam, Tomotaka Kuwahara, and Mehdi Soleimanifar. Sample-efficient learning of interacting quantum systems. *Nat. Phys.*, 17(8):931–935, 2021.

[22] Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 211–220, 2008.

[23] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[24] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

[25] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32:7413–7424, 2019.

[26] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8141–8150, Red Hook, NY, USA, 2019. Curran Associates Inc.

[27] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[28] Andrew Arrasmith, M. Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J. Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 10 2021.

[29] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *arXiv preprint arXiv:1905.10415*, 2019.

[30] Srinivasan Arunachalam, Alex B. Grilo, and Henry Yuen. Quantum statistical query learning, 2020.

[31] Dave Bacon, Isaac L. Chuang, and Aram W. Harrow. Efficient quantum circuits for Schur and Clebsch-Gordan transforms. *Phys. Rev. Lett.*, 97:170502, 10 2006.

[32] Dave Bacon, Isaac L. Chuang, and Aram W. Harrow. The quantum Schur transform: I. efficient qudit circuits, 2006.

[33] Paul Baireuther, Thomas E O'Brien, Brian Tarasinski, and Carlo WJ Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *Quantum*, 2:48, 2018.

[34] Eyal Bairey, Itai Arad, and Netanel H. Lindner. Learning a local hamiltonian from local measurements. *Phys. Rev. Lett.*, 122:020504, 1 2019.

[35] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? *arXiv preprint arXiv:1810.09102*, 2018.

[36] Andreas Bärtschi and Stephan Eidenbenz. Deterministic preparation of Dicke states. In Leszek Antoni Gąsieniec, Jesper Jansson, and Christos Levcopoulos, editors, *Fundamentals of Computation Theory*, pages 126–139, Cham, 2019. Springer International Publishing.

[37] Johannes Bausch and Felix Leditzky. Quantum codes from neural networks. *New Journal of Physics*, 22(2):023005, 2020.

[38] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nat. Commun.*, 11(1):808, 2020.

[39] A.A. Belavin, A.M. Polyakov, and A.B. Zamolodchikov. Infinite conformal symmetry in two-dimensional quantum field theory. *Nucl. Phys. B*, 241(2):333–380, 1984.

[40] Souheil Ben-Yacoub, B Fasel, and Juergen Luettin. Fast face detection using mlp and fft. In *Proc. Second International Conference on Audio and Video-based Biometric Person Authentication (AVBPA'99)*, number CONF, pages 31–36, 1999.

[41] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1):1–9, 2019.

[42] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019.

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

[44] Ingemar Bengtsson and Karol Życzkowski. *Geometry of Quantum States: An Introduction to Quantum Entanglement*. Cambridge University Press, 2017.

[45] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[46] Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha T Karhunen. Bidirectional recurrent neural networks as generative models. *Advances in Neural Information Processing Systems*, 28:856–864, 2015.

[47] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[48] Hans Bethe. Zur theorie der metalle. *Z. Phys.*, 71(3):205–226, 1931.

[49] Kishor Bharti and Tobias Haug. Quantum assisted simulator. *arXiv preprint arXiv:2011.06911*, 2020.

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

[51] Alessandro Bisio, Giulio Chiribella, Giacomo Mauro D'Ariano, Stefano Facchini, and Paolo Perinotti. Optimal quantum learning of a unitary transformation. *Physical Review A*, 81(3):032324, 2010.

[52] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018.

[53] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 253–262, New York, NY, USA, 1994. Association for Computing Machinery.

[54] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[55] Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 161–168, Red Hook, NY, USA, 2007. Curran Associates Inc.

[56] Fernando G. S. L. Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances, 2018.

[57] Fernando GSL Brandao and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426. IEEE, 2017.

[58] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017.

[59] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[60] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, 9 2001.

[61] Vivien Cabannes, Bobak T Kiani, Randall Balestriero, Yann LeCun, and Alberto Bietti. The ssl interplay: Augmentations, inductive bias, and generalization. *arXiv preprint arXiv:2302.02774*, 2023.

[62] Chris Cade, Lana Mineh, Ashley Montanaro, and Stasja Stanisic. Strategies for solving the fermi-hubbard model on near-term quantum computers. *Phys. Rev. B*, 102:235122, 12 2020.

[63] Emanuele Caglioti, François Golse, and Thierry Paul. Towards optimal transport for quantum densities. preprint, Dec 2018.

[64] Emanuele Caglioti, François Golse, and Thierry Paul. Quantum Optimal Transport is Cheaper. *Journal of Statistical Physics*, 2020.

[65] Ernesto Campos, Aly Nasrallah, and Jacob Biamonte. Abrupt transitions in variational quantum circuit training. *Phys. Rev. A*, 103:032607, 3 2021.

[66] Eric A Carlen and Jan Maas. An analog of the 2-Wasserstein metric in non-commutative probability under which the Fermionic Fokker–Planck equation is gradient flow for the entropy. *Communications in Mathematical Physics*, 331(3):887–926, 2014.

[67] Eric A Carlen and Jan Maas. Gradient flow and entropy inequalities for quantum Markov semigroups with detailed balance. *Journal of Functional Analysis*, 273(5):1810–1869, 2017.

[68] Eric A Carlen and Jan Maas. Non-commutative calculus, optimal transport and functional inequalities in dissipative quantum systems. *Journal of Statistical Physics*, 178(2):319–378, 2020.

[69] Matthias C Caro, Hsin-Yuan Huang, Marco Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J Coles. Generalization in quantum machine learning from few training data. *Nat. Commun.*, 13(1):4919, 2022.

[70] Jacques Carolan, Masoud Mohseni, Jonathan P Olson, Mihika Prabhu, Changchen Chen, Darius Bunandar, Murphy Yuezhen Niu, Nicholas C Harris, Franco NC Wong, Michael Hochberg, et al. Variational quantum unsampling on a quantum photonic processor. *Nature Physics*, 16(3):322–327, 2020.

[71] Juan Carrasquilla, Giacomo Torlai, Roger G Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

[72] Grecia Castelazo, Quynh T Nguyen, Giacomo De Palma, Dirk Englund, Seth Lloyd, and Bobak T Kiani. Quantum algorithms for group convolution, cross-correlation, and equivariant transformations. *arXiv preprint arXiv:2109.11330*, 2021.

[73] Grecia Castelazo, Quynh T. Nguyen, Giacomo De Palma, Dirk Englund, Seth Lloyd, and Bobak T. Kiani. Quantum algorithms for group convolution, cross-correlation, and equivariant transformations. *Phys. Rev. A*, 106:032402, 9 2022.

[74] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nat. Commun.*, 12(1):1791, 2021.

[75] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

[76] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nat. Rev. Phys.*, 3(9):625–644, 2021.

[77] Marco Cerezo and Patrick J Coles. Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Sci. Technol.*, 6(3):035006, 2021.

[78] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost-function-dependent barren plateaus in shallow quantum neural networks. *arXiv preprint arXiv:2001.00550*, 2020.

[79] Shouvanik Chakrabarti, Huang Yiming, Tongyang Li, Soheil Feizi, and Xiaodi Wu. Quantum wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 6781–6792, 2019.

[80] Nicholas Chancellor. Domain wall encoding of discrete variables for quantum annealing and qaoa. *Quantum Science and Technology*, 4(4):045004, 2019.

[81] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[82] Pratik Chaudhari and Stefano Soatto. On the energy landscape of deep networks, 2017.

[83] Jin-Quan Chen, Jialun Ping, and Fan Wang. *Group Representation Theory for Physicists*. World Scientific Publishing, Singapore, 2nd edition, 2002.

[84] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4666–4677, 2018.

[85] S. Chen, A. R. Klivans, and R. Meka. Learning deep ReLU networks is fixed-parameter tractable. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 696–707, Los Alamitos, CA, USA, 2 2022. IEEE Computer Society.

[86] Senrui Chen, Sisi Zhou, Alireza Seif, and Liang Jiang. Quantum advantages for Pauli channel estimation. *Phys. Rev. A*, 105:032435, 3 2022.

[87] Sitan Chen, Aravind Gollakota, Adam R. Klivans, and Raghu Meka. Hardness of noise-free learning for two-hidden-layer neural networks, 2022.

[88] Yongxin Chen, Tryphon T Georgiou, Lipeng Ning, and Allen Tannenbaum. Matricial Wasserstein-1 distance. *IEEE control systems letters*, 1(1):14–19, 2017.

[89] Yongxin Chen, Tryphon T Georgiou, and Allen Tannenbaum. Matrix optimal mass transport: a quantum mechanical approach. *IEEE Transactions on Automatic Control*, 63(8):2612–2619, 2018.

[90] Yongxin Chen, Tryphon T Georgiou, and Allen Tannenbaum. Wasserstein geometry of quantum states and optimal transport of matrix-valued measures. In *Emerging Applications of Control and Systems Theory*, pages 139–150. Springer, 2018.

[91] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[92] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 5 2015. PMLR.

[93] William R Clements, Peter C Humphreys, Benjamin J Metcalf, W Steven Kolthammer, and Ian A Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, 2016.

[94] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR.

[95] Benoît Collins. Moments and cumulants of polynomial random variables on unitarygroups, the itzykson-zuber integral, and free probability. *Int. Math. Res. Not.*, 2003(17):953–982, 2003.

[96] Benoît Collins and Piotr Śniady. Integration with respect to the Haar measure on unitary, orthogonal and symplectic group. *Commun. Math. Phys.*, 264(3):773–795, 2006.

[97] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nat. Phys.*, 15(12):1273–1278, 8 2019.

[98] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[99] Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. The born supremacy: Quantum advantage and training of an ising born machine. *npj Quantum Information*, 6(1):1–11, 2020.

[100] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[101] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. *Proceedings of machine learning research*, 97:1517, 2019.

[102] Nilanjana Datta and Cambyse Rouzé. Relating relative entropy, optimal transport and Fisher information: A quantum HWI inequality. *Annales Henri Poincaré*, 21:2115–2150, 2020.

[103] Mark A Davenport and Justin Romberg. An overview of low-rank matrix recovery from incomplete observations. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):608–622, 2016.

[104] Pierre De Fouquieres and Sophie G Schirmer. A closer look at quantum control landscapes and their implication for control optimization. *Infinite dimensional analysis, quantum probability and related topics*, 16(03):1350021, 2013.

[105] Giacomo De Palma and Stefan Huber. The conditional Entropy Power Inequality for quantum additive noise channels. *Journal of Mathematical Physics*, 59(12):122201, 2018.

[106] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. *Advances in Neural Information Processing Systems*, 32, 2019.

[107] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Adversarial robustness guarantees for random deep neural networks. In *International Conference on Machine Learning*, pages 2522–2534. PMLR, 2021.

[108] Giacomo De Palma, Milad Marvian, Dario Trevisan, and Seth Lloyd. The Quantum Wasserstein Distance of Order 1. *IEEE Transactions on Information Theory*, 67(10):6627–6643, 2021.

[109] Giacomo De Palma, Milad Marvian, Dario Trevisan, and Seth Lloyd. The quantum wasserstein distance of order 1. *IEEE Trans. Inf. Theory*, 67(10):6627–6643, 2021.

[110] Giacomo De Palma and Dario Trevisan. Quantum optimal transport with quantum channels. *Annales Henri Poincaré*, 22(10):3199–3234, 2021.

[111] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numer. Math.*, 108(1):59–91, 2007.

[112] Abhinav Deshpande, Pradeep Niroula, Oles Shtanko, Alexey V. Gorshkov, Bill Fefferman, and Michael J. Gullans. Tight bounds on the convergence of noisy random circuits to the uniform distribution, 2021.

[113] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[114] Ilias Diakonikolas, Daniel Kane, and Nikos Zarifis. Near-optimal sq lower bounds for agnostically learning halfspaces and relus under gaussian marginals. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13586–13596. Curran Associates, Inc., 2020.

[115] Ilias Diakonikolas, Daniel M. Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for PAC learning one-hidden-layer ReLU networks. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 1514–1539. PMLR, 09–12 Jul 2020.

[116] R. H. Dicke. Coherence in spontaneous radiation processes. *Phys. Rev.*, 93:99–110, 1 1954.

[117] Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

[118] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006.

[119] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. The expressive power of parameterized quantum circuits. *arXiv preprint arXiv:1810.11922*, 2018.

[120] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parametrized quantum circuits. *Phys. Rev. Research*, 2:033125, 7 2020.

[121] Yuxuan Du, Zhuozhuo Tu, Xiao Yuan, and Dacheng Tao. Efficient measure for the expressivity of variational quantum algorithms. *Phys. Rev. Lett.*, 128:080506, 2 2022.

[122] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing, 2022.

[123] Rocco Duvenhage and Machiel Snyman. Balance between quantum Markov semigroups. *Annales Henri Poincaré*, 19(6):1747–1786, 2018.

[124] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[125] Rainer Engelken, Fred Wolf, and Larry F Abbott. Lyapunov spectra of chaotic recurrent neural networks. *arXiv preprint arXiv:2006.02427*, 2020.

[126] N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070*, 2020.

[127] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[128] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[129] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. The quantum approximate optimization algorithm and the Sherrington–Kirkpatrick model at infinite size, 2019.

[130] Mark Fingerhuth, Tomáš Babej, et al. A quantum alternating operator ansatz with hard and soft constraints for lattice protein folding. *arXiv preprint arXiv:1810.13411*, 2018.

[131] Thomas Frerix and Joan Bruna. Approximating orthogonal matrices with effective givens factorization. In *International Conference on Machine Learning*, pages 1993–2001. PMLR, 2019.

[132] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[133] William Fulton. *Young Tableaux: With Applications to Representation Theory and Geometry*. London Mathematical Society Student Texts. Cambridge University Press, Cambridge, UK, 1996.

[134] Jun Gao, Lu-Feng Qiao, Zhi-Qiang Jiao, Yue-Chi Ma, Cheng-Qiu Hu, Ruo-Jing Ren, Ai-Lin Yang, Hao Tang, Man-Hong Yung, and Xian-Min Jin. Experimental machine learning of quantum states. *Physical review letters*, 120(24):240501, 2018.

[135] Li Gao, Marius Junge, and Nicholas LaRacuente. Fisher information and logarithmic sobolev inequality for matrix-valued functions. *Annales Henri Poincaré*, 21(11):3409–3478, 2020.

[136] Rui Gao, Xi Chen, and Anton J Kleywegt. Wasserstein distributional robustness and regularization in statistical learning. *arXiv preprint arXiv:1712.06050*, 2017.

[137] Xun Gao, Eric R. Anschuetz, Sheng-Tao Wang, J. Ignacio Cirac, and Mikhail D. Lukin. Enhancing generative models via quantum correlations. *Phys. Rev. X*, 12:021037, 5 2022.

[138] Xiaozhen Ge, Haijin Ding, Herschel Rabitz, and Re-Bing Wu. Robust quantum control in games: An adversarial learning approach. *Physical Review A*, 101(5):052317, 2020.

[139] Surbhi Goel, Aravind Gollakota, Zhihan Jin, Sushrut Karmalkar, and Adam Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3587–3596. PMLR, 13–18 Jul 2020.

[140] Surbhi Goel, Aravind Gollakota, and Adam Klivans. Statistical-query lower bounds via functional gradients. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[141] Aravind Gollakota and Daniel Liang. On the Hardness of PAC-learning Stabilizer States with Noise. *Quantum*, 6:640, 2 2022.

[142] François Golse. The quantum N-body problem in the mean-field and semiclassical regime. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118):20170229, 2018.

[143] François Golse, Clément Mouhot, and Thierry Paul. On the mean field and classical limits of quantum mechanics. *Communications in Mathematical Physics*, 343(1):165–205, 2016.

[144] François Golse and Thierry Paul. The Schrödinger equation in the mean-field and semiclassical regime. *Archive for Rational Mechanics and Analysis*, 223(1):57–94, 2017.

[145] François Golse and Thierry Paul. Wave packets and the quadratic Monge–Kantorovich distance in quantum mechanics. *Comptes Rendus Mathematique*, 356(2):177–197, 2018.

[146] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[147] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[148] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[149] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018.

[150] Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nat. Commun.*, 10(1):3007, 2019.

[151] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *arXiv preprint arXiv:2008.07669*, 2020.

[152] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[153] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[154] Shouzhen Gu, Rolando D. Somma, and Burak Şahinoğlu. Fast-forwarding quantum evolution. *Quantum*, 5:577, 11 2021.

[155] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[156] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

[157] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019.

[158] Jonas Haferkamp. Random quantum circuits are approximate unitary t-designs in depth $o\left(nt^{5+o(1)}\right)$, 2022.

[159] Brian Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer, 2015.

[160] Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv preprint arXiv:1408.2873*, 2014.

[161] Aram Harrow and Saeed Mehraban. Approximate unitary t-designs by short random quantum circuits using nearest-neighbor and long-range gates, 2018.

[162] Vojt ěch Havlíček and Sergii Strelchuk. Quantum Schur sampling circuits can be strongly simulated. *Phys. Rev. Lett.*, 121:060505, 8 2018.

[163] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[164] W. Heisenberg. Zur theorie des ferromagnetismus. *Z. Phys.*, 49(9):619–636, 1928.

[165] Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *International Conference on Machine Learning*, pages 1969–1978. PMLR, 2018.

[166] Mikael Henaff, Arthur Szlam, and Yann LeCun. Recurrent orthogonal networks and long-memory tasks. In *International Conference on Machine Learning*, pages 2034–2042. PMLR, 2016.

[167] Daniel Herr, Benjamin Obert, and Matthias Rosenkranz. Anomaly detection with variational quantum generative adversarial networks. *arXiv preprint arXiv:2010.10492*, 2020.

[168] Nicholas J Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM review*, 51(4):747–764, 2009.

[169] Marcel Hinsche, Marios Ioannou, Alexander Nietner, Jonas Haferkamp, Yihui Quek, Dominik Hangleiter, Jean-Pierre Seifert, Jens Eisert, and Ryan Sweke. Learnability of the output distributions of local quantum circuits, 2021.

[170] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[171] Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the quantum alternating operator ansatz. *arXiv preprint arXiv:1911.05296*, 2019.

[172] Emiel Hoogeboom, Victor Garcia Satorras, Jakub M Tomczak, and Max Welling. The convolution exponential and generalized sylvester flows. *arXiv preprint arXiv:2006.01910*, 2020.

[173] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, et al. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[174] Bing Huang, Nadine O Symonds, and O Anatole von Lilienfeld. Quantum machine learning in chemistry and materials. *Handbook of Materials Modeling: Methods: Theory and Modeling*, pages 1883–1909, 2020.

[175] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning. *Nat. Commun.*, 12(1):2631, 5 2021.

[176] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nat. Phys.*, 16(10):1050–1057, 2020.

[177] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Efficient estimation of pauli observables by derandomization. *arXiv preprint arXiv:2103.07510*, 2021.

[178] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Information-theoretic bounds on quantum advantage in machine learning. *Phys. Rev. Lett.*, 126:190505, 5 2021.

[179] Lei Huang, Li Liu, Fan Zhu, Diwen Wan, Zehuan Yuan, Bo Li, and Ling Shao. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6429–6438, 2020.

[180] Patrick Huembeli and Alexandre Dauphin. Characterizing the loss landscape of variational quantum circuits. *arXiv preprint arXiv:2008.02785*, 2020.

[181] Stephanie Hyland and Gunnar Rätsch. Learning unitary operators with help from u (n). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[182] Kazuki Ikeda. Foundation of quantum optimal transport and applications. *Quantum Information Processing*, 19(1):25, 2020.

[183] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*, 2015.

[184] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[185] Tiefeng Jiang. Maxima of entries of haar distributed matrices. *Probab. Theory Relat. Fields*, 131(1):121–144, 2005.

[186] Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pages 1733–1741. PMLR, 2017.

[187] Peter D Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and Alán Aspuru-Guzik. Qvector: an algorithm for device-tailored quantum error correction. *arXiv preprint arXiv:1711.02249*, 2017.

[188] Tyson Jones and Simon C Benjamin. Quantum compilation and circuit optimisation via energy dissipation. *arXiv preprint arXiv:1811.03147*, 2018.

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

[190] Leonid Vitalievich Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

[191] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[192] Jaroslav Kautsky and Radka Turcajová. A matrix approach to discrete wavelets. In *Wavelet Analysis and Its Applications*, volume 5, pages 117–135. Elsevier, 1994.

[193] David Kazhdan and George Lusztig. Affine lie algebras and quantum groups. *Int. Math. Res. Not.*, 1991(2):21–29, 1991.

[194] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 11 1998.

[195] Joseph B Keller. Closest unitary, orthogonal and hermitian operators to a given operator. *Mathematics Magazine*, 48(4):192–197, 1975.

[196] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.

[197] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. Quantum-assisted quantum compiling. *Quantum*, 3:140, 5 2019.

[198] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, Hossein Sadeghi, and Mohammad H Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001, 2018.

[199] Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. projUNN: efficient method for training deep networks with unitary matrices. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[200] Bobak T. Kiani. bkiani/Beyond-Barren-Plateaus: Code for Beyond Barren Plateaus paper, 11 2022.

[201] Bobak T Kiani, Giacomo De Palma, Dirk Englund, William Kaminsky, Milad Marvian, and Seth Lloyd. Quantum advantage for differential equation analysis. *arXiv preprint arXiv:2010.15776*, 2020.

[202] Bobak Toussi Kiani, Giacomo De Palma, Dirk Englund, William Kaminsky, Milad Marvian, and Seth Lloyd. Quantum advantage for differential equation analysis. *Physical Review A*, 105(2):022415, 2022.

[203] Bobak Toussi Kiani, Giacomo De Palma, Milad Marvian, Zi-Wen Liu, and Seth Lloyd. Quantum earth mover's distance: A new approach to learning quantum data, 2021.

[204] Bobak Toussi Kiani, Giacomo De Palma, Milad Marvian, Zi-Wen Liu, and Seth Lloyd. Quantum earth mover's distance: A new approach to learning quantum data. *arXiv preprint arXiv:2101.03037*, 2021.

[205] Bobak Toussi Kiani, Giacomo De Palma, Milad Marvian, Zi-Wen Liu, and Seth Lloyd. Learning quantum data with the quantum earth mover's distance. *Quantum Science and Technology*, 7(4):045002, 2022.

[206] Bobak Toussi Kiani, Seth Lloyd, and Reevu Maity. Learning unitaries by gradient descent. *arXiv preprint arXiv:2001.11897*, 2020.

[207] Bobak Toussi Kiani, Agnes Villanyi, and Seth Lloyd. Quantum medical imaging algorithms. *arXiv preprint arXiv:2004.02036*, 2020.

[208] Mária Kieferová and Nathan Wiebe. Tomography and generative training with quantum Boltzmann machines. *Phys. Rev. A*, 96:062327, 12 2017.

[209] Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

[210] Joonho Kim, Jaedeok Kim, and Dario Rosa. Universal effectiveness of high-depth circuits in variational eigenproblems, 2020.

[211] Joonho Kim and Yaron Oz. Quantum energy landscape and VQA optimization, 2021.

[212] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[213] Alexander Kirillov Jr. *An introduction to Lie groups and Lie algebras*. Number 113. Cambridge University Press, 2008.

[214] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871. PMLR, 2014.

[215] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[216] Martin Larocca, Esteban A Calzetta, and Diego A Wisniacki. Navigating on quantum control solution subspaces. *arXiv preprint arXiv:2001.05941*, 2020.

[217] Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J. Coles, and M. Cerezo. Theory of overparametrization in quantum neural networks, 2021.

[218] Martín Larocca, Frédéric Sauvage, Faris M. Sbahi, Guillaume Verdon, Patrick J. Coles, and M. Cerezo. Group-invariant quantum machine learning. *PRX Quantum*, 3:030341, Sep 2022.

[219] Hannah Lawrence, Kristian Georgiev, Andrew Dienes, and Bobak T. Kiani. Implicit bias of linear equivariant networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12096–12125. PMLR, 17–23 Jul 2022.

[220] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[221] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[222] M. A. Levin and X.-G. Wen. String-net condensation: A physical mechanism for topological phases. *Phys. Rev. B*, 71:045110, 2005.

[223] Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pages 3794–3803. PMLR, 2019.

[224] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[225] Jun Li, Li Fuxin, and Sinisa Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. *arXiv preprint arXiv:2002.01113*, 2020.

[226] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Advances in neural information processing systems*, 32:15390–15402, 2019.

[227] Shuai Li, Kui Jia, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019.

[228] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8168–8177, Red Hook, NY, USA, 2018. Curran Associates Inc.

[229] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47. PMLR, 2018.

[230] Zefeng Li, Men-Andrin Meier, Egill Hauksson, Zhongwen Zhan, and Jennifer Andrews. Machine learning seismic wave discrimination: Application to earthquake early warning. *Geophysical Research Letters*, 45(10):4773–4779, 2018.

[231] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

[232] Junyu Liu, Khadijeh Najafi, Kunal Sharma, Francesco Tacchino, Liang Jiang, and Antonio Mezzacapo. An analytic theory for the dynamics of wide quantum neural networks, 2022.

[233] Weiyang Liu, Rongmei Lin, Zhen Liu, James M Rehg, Liam Paull, Li Xiong, Le Song, and Adrian Weller. Orthogonal over-parameterized training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7251–7260, 2021.

[234] Yi-Kai Liu. Universal low-rank matrix recovery from pauli measurements. *Advances in Neural Information Processing Systems*, 24:1638–1646, 2011.

[235] Seth Lloyd. Quantum approximate optimization is computationally universal. *arXiv preprint arXiv:1812.11075*, 2018.

[236] Seth Lloyd, Samuel Bosch, Giacomo De Palma, Bobak Kiani, Zi-Wen Liu, Milad Marvian, Patrick Rebentrost, and David M Arvidsson-Shukur. Quantum polar decomposition algorithm. *arXiv preprint arXiv:2006.00841*, 2020.

[237] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[238] Guang Hao Low. Classical shadows of fermions with particle number symmetry, 2022.

[239] Sirui Lu, Lu-Ming Duan, and Dong-Ling Deng. Quantum adversarial machine learning. *Physical Review Research*, 2(3):033212, 2020.

[240] Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert J. N. Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[241] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *Phys. Rev. A*, 103:012405, 1 2021.

[242] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement induced barren plateaus, 2020.

[243] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*, 2018.

[244] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

[245] Michael Mathieu and Yann LeCun. Fast approximation of rotations and hessians matrices. *arXiv preprint arXiv:1404.7195*, 2014.

[246] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.*, 9(1):4812, 2018.

[247] Johannes Jakob Meyer, Marian Mularski, Elies Gil-Fuster, Antonio Anna Mele, Francesco Arzani, Alissa Wilms, and Jens Eisert. Exploiting symmetry in variational quantum machine learning, 2022.

[248] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[249] Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pages 2401–2409. PMLR, 2017.

[250] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, 9 2018.

[251] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[252] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[253] Katharine W Moore and Herschel Rabitz. Exploring constrained quantum control landscapes. *The Journal of chemical physics*, 137(13):134113, 2012.

[254] Kouhei Nakaji and Naoki Yamamoto. Quantum semi-supervised generative adversarial network for enhanced data classification. *arXiv preprint arXiv:2010.13727*, 2020.

[255] Kouhei Nakaji and Naoki Yamamoto. Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, 5:434, 4 2021.

[256] John Napp. Quantifying the barren plateau phenomenon for a model of unstructured variational ansätze, 2022.

[257] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J Briegel, and Nicolai Friis. Optimizing quantum error correction codes with reinforcement learning. *Quantum*, 3:215, 2019.

[258] Behnam Neyshabur, Srinadh Bhojanapalli, David Mcallester, and Nati Srebro. Exploring generalization in deep learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[259] Quynh T Nguyen, Bobak T Kiani, and Seth Lloyd. Block-encoding dense and full-rank kernels using hierarchical matrices: applications in quantum numerical linear algebra. *Quantum*, 6:876, 2022.

[260] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[261] Yasunori Nishimori and Shotaro Akaho. Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[262] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2022. Published electronically at http://oeis.org, Sequence A000292.

[263] Peter J. Olver. *Classical Invariant Theory*. London Mathematical Society Student Texts. Cambridge University Press, Cambridge, UK, 1999.

[264] Donald S Ornstein. An application of ergodic theory to probability theory. *The Annals of Probability*, 1(1):43–58, 1973.

[265] Roman Orus, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[266] Pantita Palittapongarnpim, Peter Wittek, Ehsan Zahedinejad, Shakib Vedaie, and Barry C Sanders. Learning in quantum control: High-dimensional global optimization for noisy quantum dynamics. *Neurocomputing*, 268:116–126, 2017.

[267] Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

[268] Robert M Parrish, Edward G Hohenstein, Peter L McMahon, and Todd J Martínez. Quantum computation of electronic transitions using a variational quantum eigensolver. *Physical review letters*, 122(23):230401, 2019.

[269] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[270] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[271] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[272] Alexander N Pechen and David J Tannor. Are there traps in quantum control landscapes? *Physical review letters*, 106(12):120402, 2011.

[273] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.*, 5(1):4213, 2014.

[274] Arthur Pesah, M Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *arXiv preprint arXiv:2011.02966*, 2020.

[275] Arthur Pesah, M. Cerezo, Samson Wang, Tyler Volkoff, Andrew T. Sornborger, and Patrick J. Coles. Absence of barren plateaus in quantum convolutional neural networks. *Phys. Rev. X*, 11:041011, 10 2021.

[276] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894*, 2017.

[277] Gabriel Peyré and Marco Cuturi. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[278] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 8 2018.

[279] Omri Puny, Derek Lim, Bobak T Kiani, Haggai Maron, and Yaron Lipman. Equivariant polynomials for graph neural networks. *arXiv preprint arXiv:2302.11556*, 2023.

[280] Marco Túlio Quintino, Qingxiuxiong Dong, Atsushi Shimbo, Akihito Soeda, and Mio Murao. Reversing unknown quantum transformations: Universal quantum circuit for inverting general unitary operations. *Physical Review Letters*, 123(21):210502, 2019.

[281] Giulio Racah. Theory of complex spectra. II. *Phys. Rev.*, 62:438–462, 11 1942.

[282] Michael Ragone, Paolo Braccia, Quynh T Nguyen, Louis Schatzki, Patrick J Coles, Frederic Sauvage, Martin Larocca, and M Cerezo. Representation theory for geometric quantum machine learning, 2022.

[283] Lev Reyzin. Statistical queries and statistical algorithms: Foundations and applications, 2020.

[284] Andrea Rocchetto, Scott Aaronson, Simone Severini, Gonzalo Carvacho, Davide Poderini, Iris Agresti, Marco Bentivegna, and Fabio Sciarrino. Experimental learning of quantum states. *Science advances*, 5(3):eaau1946, 2019.

[285] Jonathan Romero and Alan Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *arXiv preprint arXiv:1901.00848*, 2019.

[286] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, 2017.

[287] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in neural information processing systems*, pages 2018–2028, 2017.

[288] Cambyse Rouzé and Nilanjana Datta. Concentration of quantum states from quantum functional and transportation cost inequalities. *Journal of Mathematical Physics*, 60(1):012202, 2019.

[289] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[290] Ernest K Ryu, Yongxin Chen, Wuchen Li, and Stanley Osher. Vector and matrix optimal mass transport: theory, algorithm, and applications. *SIAM Journal on Scientific Computing*, 40(5):A3675–A3698, 2018.

[291] Jun John Sakurai and Eugene D Commins. Modern quantum mechanics, revised edition, 1995.

[292] Louis Schatzki, Martin Larocca, Quynh T. Nguyen, Frederic Sauvage, and M. Cerezo. Theoretical guarantees for permutation-equivariant quantum neural networks, 2022.

[293] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99:032331, 3 2019.

[294] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[295] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[296] Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

[297] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3067–3075. JMLR.org, 2017.

[298] Kunal Sharma, Marco Cerezo, Lukasz Cincio, and Patrick J Coles. Trainability of dissipative perceptron-based quantum neural networks. *arXiv preprint arXiv:2005.12458*, 2020.

[299] Kunal Sharma, Sumeet Khatri, Marco Cerezo, and Patrick J Coles. Noise resilience of variational quantum compiling. *New Journal of Physics*, 22(4):043006, 2020.

[300] Huitao Shen, Pengfei Zhang, Yi-Zhuang You, and Hui Zhai. Information scrambling in quantum neural networks. *Phys. Rev. Lett.*, 124:200504, 5 2020.

[301] Yichen Shen, Nicholas C Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, et al. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7):441, 2017.

[302] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[303] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Adv. Quantum Technol.*, 2(12):1900070, 2019.

[304] Sahil Singla and Soheil Feizi. Skew orthogonal convolutions. *arXiv preprint arXiv:2105.11417*, 2021.

[305] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *arXiv preprint arXiv:2006.14904*, 2020.

[306] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Mach. Intell.*, 3(1):5, 2021.

[307] Rolando Somma, Howard Barnum, Gerardo Ortiz, and Emanuel Knill. Efficient solvability of Hamiltonians and limits on the power of some quantum computational models. *Phys. Rev. Lett.*, 97:190501, 11 2006.

[308] Nikitas Stamatopoulos, Daniel J Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option pricing using quantum computers. *Quantum*, 4:291, 2020.

[309] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum Natural Gradient. *Quantum*, 4:269, 5 2020.

[310] Bernd Sturmfels. *Algorithms in Invariant Theory.* Texts & Monographs in Symbolic Computation. Springer Vienna, Vienna, Austria, 2008.

[311] Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 398:185–196, 2020.

[312] Balázs Szörényi. Characterizing statistical query learning: Simplified notions and proofs. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 186–200, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[313] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

[314] Giacomo Torlai and Roger G Melko. Machine-learning quantum states in the nisq era. *Annual Review of Condensed Matter Physics*, 11:325–344, 2020.

[315] Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. *arXiv preprint arXiv:2104.07167*, 2021.

[316] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 11 1984.

[317] Joran Van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. *Quantum*, 4:230, 2020.

[318] Tan Van Vu and Yoshihiko Hasegawa. Geometrical Bounds of the Irreversibility in Markovian Systems. *arXiv preprint arXiv:2005.02871*, 2020.

[319] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[320] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. A quantum algorithm to train neural networks using low-depth circuits. *arXiv preprint arXiv:1712.05304*, 2017.

[321] Anatoly Moiseevich Vershik. Long history of the Monge-Kantorovich transportation problem. *The Mathematical Intelligencer*, 35(4):1–9, 2013.

[322] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[323] Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[324] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. 2019.

[325] Dan Voiculescu. Limit laws for random matrices and free products. *Invent. Math.*, 104(1):201–220, 1991.

[326] John Von Neumann. *Mathematical foundations of quantum mechanics: New edition*, volume 53. Princeton university press, 2018.

[327] Daochen Wang, Oscar Higgott, and Stephen Brierley. Accelerated variational quantum eigensolver. *Physical review letters*, 122(14):140504, 2019.

[328] Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11505–11515, 2020.

[329] Samson Wang, Enrico Fontana, Marco Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J Coles. Noise-induced barren plateaus in variational quantum algorithms. *arXiv preprint arXiv:2007.14384*, 2020.

[330] Xin Wang, Zhixin Song, and Youle Wang. Variational Quantum Singular Value Decomposition. *Quantum*, 5:483, 6 2021.

[331] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2):022304, 2018.

[332] Zirui Wang, Jun Wang, and Youren Wang. An intelligent diagnosis scheme based on generative adversarial learning deep neural networks and its application to planetary gearbox fault pattern recognition. *Neurocomputing*, 310:213–222, 2018.

[333] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Phys. Rev. A*, 92:042303, Oct 2015.

[334] Dave Wecker, Matthew B Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Physical Review A*, 92(4):042303, 2015.

[335] Roeland Wiersema, Cunlu Zhou, Yvette de Sereville, Juan Felipe Carrasquilla, Yong Baek Kim, and Henry Yuen. Exploring entanglement and optimization within the hamiltonian variational ansatz. *PRX Quantum*, 1:020319, 12 2020.

[336] Scott Wisdom, Thomas Powers, John R Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. *arXiv preprint arXiv:1611.00035*, 2016.

[337] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1(1):67–82, 1997.

[338] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1):4–24, 2021.

[339] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR, 2018.

[340] Qi Xuan, Zhuangzhi Chen, Yi Liu, Huimin Huang, Guanjun Bao, and Dan Zhang. Multiview generative adversarial network and its application in pearl classification. *IEEE Transactions on Industrial Electronics*, 66(10):8244–8252, 2018.

[341] Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, et al. Quantum image processing and its application to edge detection: theory and experiment. *Physical Review X*, 7(3):031041, 2017.

[342] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[343] Xuchen You, Shouvanik Chakrabarti, and Xiaodi Wu. A convergence theory for over-parameterized variational quantum eigensolvers, 2022.

[344] Xuchen You, Shouvanik Chakrabarti, and Xiaodi Wu. A convergence theory for over-parameterized variational quantum eigensolvers, 2022.

[345] Xuchen You and Xiaodi Wu. Exponentially many local minima in quantum neural networks. In *International Conference on Machine Learning*, pages 12144–12155. PMLR, 2021.

[346] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.

[347] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, 2019.

[348] A. Yu. Zaitsev. Estimates for the Levy-Prokhorov distance in terms of characteristic functions and some of their applications. *J. Sov. Math.*, 27(5):3070–3083, 1984.

[349] Robert Zeier and Thomas Schulte-Herbrüggen. Symmetry principles in quantum systems theory. *J. Math. Phys.*, 52(11):113510, 2011.

[350] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and inference on generative adversarial quantum circuits. *Physical Review A*, 99(5):052306, 2019.

[351] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, 2 2021.

[352] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[353] Yuxuan Zhang, Ruizhe Zhang, and Andrew C Potter. Qed driven qaoa for network-flow optimization. *arXiv preprint arXiv:2006.09418*, 2020.

[354] Chen Zhao and Xiao-Shan Gao. Analyzing the barren plateau phenomenon in training quantum neural networks with the zx-calculus. *Quantum*, 5:466, 2021.

[355] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.

[356] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.

[357] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Variational quantum Boltzmann machines. *Quantum Mach. Intell.*, 3(1):7–21, 2021.

[358] Karol Zyczkowski and Wojciech Slomczynski. The Monge metric on the sphere and geometry of quantum states. *Journal of Physics A: Mathematical and General*, 34(34):6689, 2001.

[359] Karol Zyczkowski and Wojeciech Slomczynski. The Monge distance between quantum states. *Journal of Physics A: Mathematical and General*, 31(45):9095, 1998.