Advances in quantum machine learning

By

MAX WILSON

Department of Engineering
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

MARCH 2021

Abstract

Quantum machine learning is a broad term encompassing machine learning algorithms that are either partly or entirely based on quantum information processing principles or entirely classical machine learning algorithms used to solve quantum mechanical problems.

Here, we explore three realizations of quantum machine learning: a hybrid quantum-classical generative adversarial network, hybrid quantum-classical variational algorithms, and finally classical fermionic neural network Ansatz used in quantum Monte Carlo. In the first, we develop the first implementation of a quantum-classical generative adversarial network, showing the first, to our knowledge, known application to a hybrid quantum-classical model to a complex color dataset. In the second, we demonstrate how classical machine learning algorithms, specifically metalearning, can be advantageous to integrate with novel variational quantum algorithms for optimization showing that metalearning methods have better performance than other commonly used methods especially in the presence of parameter setting noise. Finally, in the third, we extend existing work on variational Monte Carlo with fermionic neural network Ansatz by improving the network design and further propagating the wave function with diffusion Monte Carlo, achieving state-of-the-art performance on some small atomic systems (Be-Ne, C^+).

ACKNOWLEDGEMENTS

Over the last four years I have written over 100,000 of code (to varying degrees of quality), travelled around 50,000 kilometers, written over a thousand pages, cried 7 times, met hundreds of talented people, burned enough compute to make me concerned for the polar bears, fell in-and-out of love, and learned some things. Yes the error bars on these estimates are large and no I did not do any of this alone. Well, some of it was done alone, but you get the idea. Hopefully I can do the fabulous people in my life justice with a shout-out here. If you read this and you're disappointed you are not immortalised in these words, well you should have been a better friend (jk call me I'll fix it, but let's be honest who is even reading this anyway).

Thanks to my family Mitz, Weathered Wolf, Fats and Alison for the constant loving support, being great human beings, and making me laugh. You make me aspire to be a better person, for example I hope that one day I too can declare "I have no thoughts". To my nieces Flo and Eve for existing, if you read this and provide me the access code gr33n-flamingo you will be financially rewarded. To Grandma Maxine for teaching me maths all those years ago.

To my friends you get some letters for all the good times, the memories and the free therapy: AA, JB, AF, MN, ED, AS, KK, JM, SV, JF, JM, FW, JW, EM, BF, SC, RN, LP, JS, KM, MG, NG, _lamingo. In particular, thanks to Mae, who helped me understand myself more than physics (even though she knows a s*** tonne): lybbly. No you can't all have named shout-outs.

To the team at Bristol and Cohort 3, something something team work makes the dream work. Especially to Jorge whose words "there's this f****** thing in my brain and it keeps me going", inspired me to not follow my dreams and move to Paris to be an actor. Not sure if this one is a thanks or not. To Colin Campbell for taking me on when no one else would and putting up with the administrative burden of an astray student, it is only through your kindness that I was able to do the research I wanted to do in the places I needed to be.

To the team at NASA Ames for being welcoming, full of great ideas, and always open to discuss their expertise. Special and sincere thanks to Eleanor Rieffel whose positivity, love of research, genuine kindness and excellent guidance I was lucky enough to experience. To the friends I made in California who made it feel like home and provided motivation when times were rough, in the words of Jason "I'm (now) bettin' on me, baby".

Thanks to $\phi\phi$, an exceptional collaborator, with talent up to his eye balls, who provided feedback on some of the Chapters. That said, it is not clear to what extent Jula is the true Markovian dynamics virtuoso in that relationship. Either way, the most important thing is who is the current Lord of Catan (Zorro).

The world is full of fabulous people and I am very lucky to have interacted with so many over the course my Ph.D. You rock. That's all folks, or as we say in Polish, "Siema, jestem Adam..."...

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: DATE: 25TH MARCH 2021

PUBLICATIONS

R. Serban, **M. Wilson**, M. Benedetti, J. Realpe-Gomez, A. Perdomo-Ortiz, A. Petukov and P. Jayakumar, *Quantum annealing for mobility studies: Go/no-go maps via quantum assisted machine learning*, in Proceedings of GVSETS conference in the Modeling & Simulation, Testing & Validation track, 2018

M. Wilson, T. Vandal, T. Hogg and E. Rieffel, *Quantum-assisted associative adversarial network: Applying quantum annealing in deep learning*, arXiv:1904.10573, 2019 (Quantum Machine Intelligence, accepted conditional on minor revisions, 2021)

M. Wilson, R. Stromswold, F. Wudarski, S. Hadfield, N. Tubman, E. Rieffel, *Optimizing quantum heuristics with meta learning*, Quantum Machine Intelligence, 2020

N. Gao, **M. Wilson**, T. Vandal, W. Vinci, R. Nemani and E. Rieffel, *High-Dimensional Similarity Search with Quantum-Assisted Variational Autoencoder*, in Proceedings of SigKDD international conference, 2020

M. Wilson, N. Gao, F. Wudarski, E. Rieffel and N. Tubman, *Simulations of state-of-the-art fermionic neural network wave functions with diffusion Monte Carlo*, arXiv:2103.12570, 2021 (Planned submission in Physical Review X)

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

In the beginning the Universe was created.
This has made a lot of people very angry and been widely regarded as a bad move.
— Douglas Adams*, The Ultimate Hitchhiker's Guide To The Galaxy*

Analysing and applying large amounts of data is a hard problem. Data is usually a set of numerical values describing an object or objects. For example, if we measure the temperature in Bristol (UK) every month for a 100 years and analyse the data (effectively), we might find that the temperature increases during summer, decreases during winter, and, though it may *not* be warm on average in any given year, there may be a concerning overall upward trend, Figure 1.1. We then can apply this understanding to timing our winter clothing purchases or deciding when to move our homes further inland. This problem of analysing and applying data becomes increasingly difficult, roughly speaking, as the dimensionality (number of variables) and size of the dataset increase [127].

Machine learning algorithms are tools for the analysis and application of data. A significant proportion of machine learning algorithms can be defined, at a high level, as the optimisation of a parameterised function, which is referred to as a model, via the processing of data. This fundamental description spans the many manifestations of machine learning, which are usually grouped under the umbrellas of 'supervised', 'unsupervised' and 'reinforcement' learning paradigms. These subfields are defined by the general characteristics of the algorithms, for example, supervised learning typically involves a **prediction** or **classification** problem, unsupervised learning attempts to **discover patterns in data**, and reinforcement learning creates models that depend on a **reward signal**.

Deep learning is the current dominant method for solving machine learning problems. It consists of different types of models (neural networks), optimisation routines and computational frameworks for analysing and applying data. These methods have been remarkably successful at solving a range of hard problems, including the protein folding problem [121], games [239] and

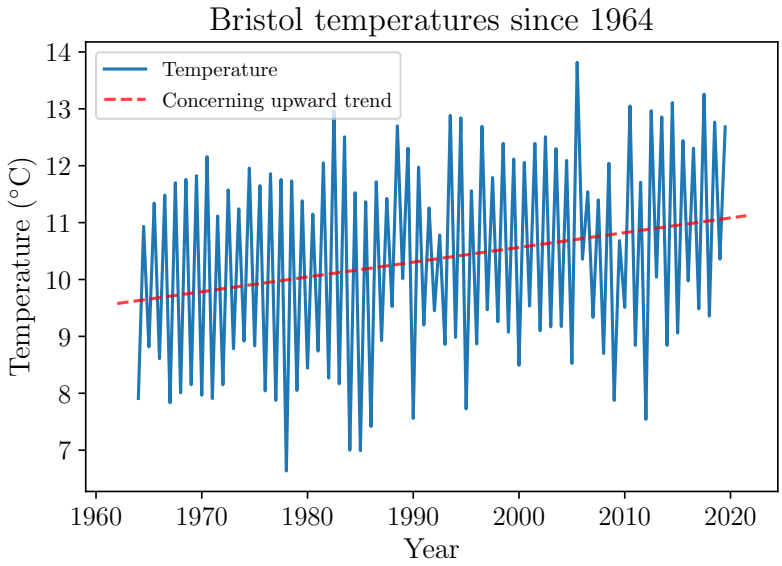


Figure 1.1: Simple analysis of the dataset [271], indicating seasonal oscillations in temperature and an overall upward trend.

physics [75], broadly resulting from the flexibility and modelling capacity of the networks [153]. These ideas have been adapted to quantum computational frameworks and applied to quantum mechanical problems [51, 108, 204].

Quantum mechanics is a set of rules that describe the behaviour of experiments. These rules predict, to a very high precision, the behaviour of some observable physics. There are, of course, short-comings (usually when describing very large systems), which are not relevant to the work in this Thesis.

Systems that behave in a quantum way, for example exhibiting wave-like nature (superposition) and uniquely quantum correlations (entanglement), are called quantum systems. A quantum system can be a collection of particles (atoms and electrons) and their relevant properties. Abstract models of quantum systems, for example sets of two level systems with properties of entanglement and superposition, can be used as a framework to develop algorithms that exploit the peculiarities of quantum mechanics to process information.

Quantum machine learning is a broad term encompassing machine learning algorithms that are **either partly or entirely based on quantum information processing principles** or **entirely classical machine learning algorithms used to solve quantum mechanical problems**, where 'classical' is used to indicate algorithms which *do not* use any quantum mechanical principles to process information. Quantum information processing is a well-established theoretical framework for processing information modelled by quantum mechanics. Recent and rapid advances in quantum information processing devices have contributed to a surge of interest in the design and application of this technology, including in the field of quantum machine

learning. Further, developments in classical machine learning, particularly with regards to the capacity of models and the computational power of GPU hardware, have encouraged researchers to apply these algorithms to quantum mechanical problems.

This Thesis explores the different ways that classical neural network theory and quantum mechanics, including quantum computation, interact. For example on the one hand, quantum mechanical systems are remarkably hard to simulate because of their quantum properties, and experiments conducted in this Thesis indicate that classical neural network methods may, in some use cases, take precedence over incumbent methods for the modelling of quantum states. On the other hand, it is these quantum properties that motivate researchers, by way of computational promise, to explore the possibility of using quantum mechanical systems (quantum computers) in neural network algorithms.

In the following Sections, quantum machine learning is motivated. First, the quantum many body problem is discussed, followed by a presentation of neural networks as models for this problem. Second, quantum machine learning is motivated and hybrid quantum-classical models based on neural network methods are introduced.

1.1 Challenges of quantum machine learning

1.1.1 Quantum many body problem

It is important to understand quantum systems and be able to make predictions about their behaviour, for reasons ranging from understanding fundamental physics and the limitations of the rules to practical problems such as designing new formula for batteries. This is not a trivial problem: The dimension of the quantum system to be modelled is exponential in the degrees of freedom. Exact modelling of a quantum system quickly becomes impractical. This problem is referred to as the quantum many-body problem.

Observing a phenomenon that we would like to understand, a physicist might conjure a simplified model of the system hoping to capture most of the physics, which will then accurately predict the phenomena. If the model accurately predicts the phenomena; it is a good model. The model is a set of equations describing the behaviour of the system. In order to understand the system, one must solve the equations. In the case of a quantum system, we must first create an accurate description of the quantum state, the object that describes the quantum system.

Accurate descriptions of quantum states allow us to predict the value of physical properties of a quantum system and to determine behaviour of the system in time. There are (effectively) uncountable classical approximations for modelling quantum systems. These methods have been designed, refined and specialised over decades and work in a range of problems, situations and computational constraints. Low energy states are of particular interest because these states frequently appear in nature.

In general, finding good descriptions of ground states (or other low energy states) of quantum systems is effectively impossible analytically and can be computationally demanding to find via heuristic methods. A heuristic method is roughly defined as a method that does not have guarantees on the quality of a solution or on the time required for a particular problem. The size of the state space of the quantum system is exponential in the number of degrees of freedom, where a degree of freedom might be the spin or orbital angular momentum of a particle, therefore scaling to larger systems can be expensive. The corollary of this is the amount of information required to describe a quantum state is also exponential in the number of degrees of freedom. To simply highlight this problem, if a small quantum system has 50 discrete degrees of freedom (each of which has 2 valid classical states), a complete description of the quantum state will require 1,125,899,906,842,624 numbers (although in practical systems many of these numbers will be effectively zero, as the state is concentrated in some smaller region of the space). Using 32-bit precision floats for these numbers results in 5 petabytes of data, for context Summit, one of the largest supercomputers in the world, has in total around 10 petabytes of RAM [141].

The importance and difficulty of the problem, simulating quantum states, stimulates a rich area of research. There is a smorgasbord of methods for finding ground states of quantum systems including Quantum Monte Carlo [85], Tensor Networks [193], Density Functional Theory [17] and Coupled-Cluster [139]. These methods work in a variety of circumstances, for example Tensor Networks are particularly well suited to quantum systems whose ground states satisfy area law entanglement. The less computationally demanding approaches, for example Density Functional Theory, can require a fair amount of tuning and intuition in their application to a particular system.

It is precisely the same properties that make quantum states difficult to simulate, notably the exponential size of the space that they are described in, that motivates the development of quantum information processing devices that can exploit these quantum properties for algorithmic speedups. These devices and algorithms are introduced in Section 1.1.3.

1.1.2 Classical models

Neural networks are conceptually derived from observations of the central nervous system. At a high level, they are connected nodes of a graph where the value of a node is a function of all the inputs to that node. In some frameworks the node will ‘fire’ (output a value) in the same way that a neuron in the central nervous system can be thought to fire after the inputs cross some threshold (though biological neurons fire in discrete spikes, there are analogous models of this behaviour from neural networks such as Bolzmann machines). They are used to represent models, which can be used to analyse data. These models are non-linear parameterised functions which can change the parameterisation to be useful for some task, or, borrowing language again from the cognitive sciences, ‘learn’. They and associated optimisation methods form the core of recent spectacular advances in machine learning, and are the focus of this Thesis.

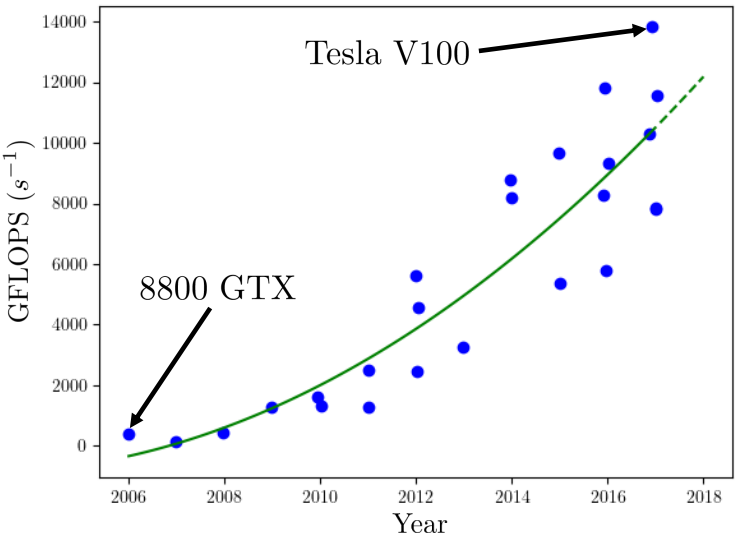


Figure 1.2: Plot showing the general trend of the computational power of GPUs since 2006 in Giga FLOPS (GFLOPS). Green line is a by eye fit of a quadratic line. Data collated from source [192], complete description of data and other sources [167].

The ideas powering neural networks go as far back as the 1940s [171]. In the following decades researchers and pundits vacillated between techno-optimism and despair [186], leading to fits and starts of progress in neural network research. A fundamental development required to usher in the practical advances made in recent years was backpropagation, arguably outlined by Paul Werbos in the 1970s [274]. Though much of the theory and understanding was developed before the end of the millennium, there were two more pieces required before neural network methods became the engine of innovation they are today. The first is the sheer quantity of accessible data, driven by the prevalence of computers and more directed efforts for it to be collected and made available. The second came in 2012 when AlexNet [138] completed the puzzle by increasing the efficiency neural networks by moving the computations from Central Processing Unit (CPU) to Graphics Processing Unit (GPU). The inherently parallel nature of these chips means they are predisposed to the computations required for neural networks, for example batched matrix multiplications, and the computational power of these chips is consistently rising, Figure 1.2, allowing larger and larger models to be trained, Figure 1.3.

Since then neural networks have claimed state-of-the-art performance in a multitude of tasks: Image classification [254], natural language processing [45], recommender systems [295] and games [239]. They are extremely flexible parameterised functions, with demonstrably exceptional capacity to learn from data.

It is for these reasons that they can be applied to many different problems in a range of

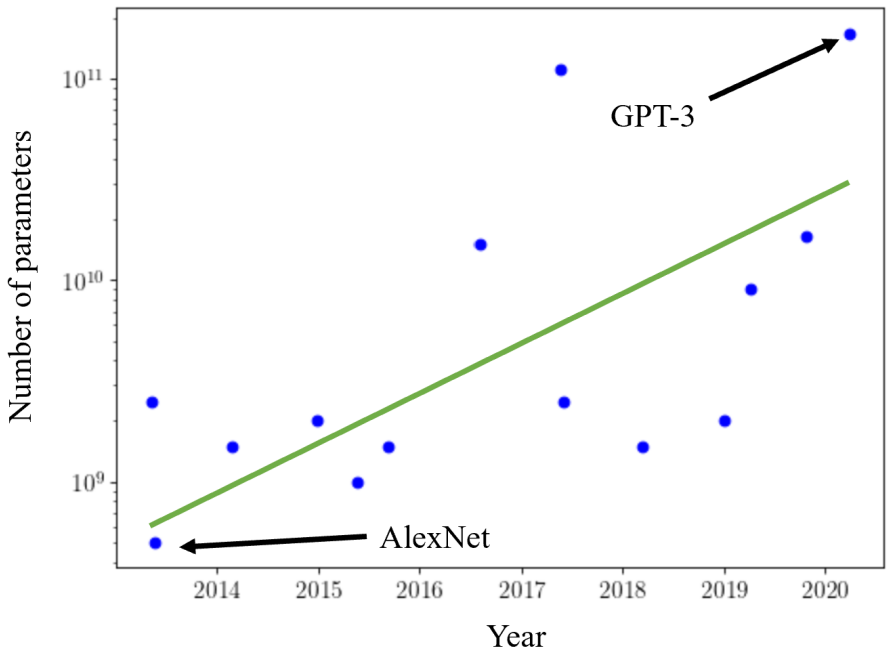


Figure 1.3: Log plot showing model size (as measured by the number of parameters) as a function of year since 2012. Green line is a log fit to the data included to highlight the general trend. Data taken from [176, 289]. AlexNet is a convolutional neural network designed for image recognition and GPT-3 is a language model producing text for a wide range of applications.

approaches. Naturally, neural network methods are beginning to be used as tools understanding quantum systems [52], have been used in a variety of ways in conjunction with quantum computers [7], and solving quantum mechanical problems [142].

1.1.3 Hybrid quantum-classical models

Although neural network algorithms have demonstrated good performance on a range of problems, they have some drawbacks. Firstly, they require large amounts of data, which is usually required to be labelled; a labour intensive process. Secondly, they are expensive to train and run, for example low-end estimates of the cost of training one of the currently largest known models at the time of writing, GPT-3, was $5 million [46]. Finally, it is not trivial to understand the behaviour of a model or predict the performance *a priori*. The first and third problems are not discussed here. However, there are theoretical speedups for quantum analogues of classical neural network algorithms, which may decrease the cost of these methods or improve the performance of hybrid quantum-classical models over their purely classical implementations.

1.1.3.1 Quantum Computing

In the 1980s researchers pointed out that quantum systems could not be efficiently simulated by classical Turing machines [84, 159, 256]. These works provided the motivation and foundational building blocks of quantum computing, the offspring of information theory and quantum mechanics [216]. Arguably, it was Shor's algorithm [238], published in the twilight of the 20th century that motivated the stunning rise of physical quantum computers. In his work, Shor presented a polynomial time quantum algorithm for factoring integers: A problem with no known classical polynomial time algorithm.

Other quantum algorithms with theoretical speedup (or improvements in performance), though it is not rigorously proven that there does not exist some faster classical algorithm, include Grover's search [97], quantum adiabatic optimisation [80], the Harrow-Hassidim-Lloyd algorithm [155] and quantum phase estimation [4], an algorithm providing exponential speedup for the problem of finding the eigenvectors and eigenvalues of a Hamiltonian. There are many other examples, explored in the reviews from References [61, 181], though often, as discussed in Reference [181], these are based on a small set of common subroutines. These algorithms come from the complexity class bounded error quantum polynomial time and running them requires some device capable of performing the required operations: quantum information processing devices or quantum computers.

There are different models of quantum computation, including quantum annealing [16], gate-model [189] and measurement-based [211]. In this Thesis, gate-model quantum computation and quantum annealing are explored, which represent the current most successful and well known physical realisations of these machines.

Physical quantum computers are difficult to build, especially at the scale (thousands or millions of qubits) required by some algorithms (reference above) due to the fragility of quantum states (they are susceptible to decoherence). As a result these algorithms may require some form of error correction [237] or quantum random access memory [40]. Recently developed devices, often referred to Noisy Intermediate Scale Quantum (NISQ) devices, are not error-corrected and suffer from large amounts of noise (decoherence, readout error, gate errors). There is much uncertainty surrounding the scalability of these devices, and it is therefore unlikely algorithms requiring thousands of error-corrected qubits will represent the first useful real-world use of quantum computation. Figures 1.4 and 1.5 show some early predictions of scaling from prominent groups in the field, for the gate-model and quantum annealing paradigms, respectively.

However, advances have been made on quantum heuristics [79, 201], algorithms with no provable and guaranteed performance bounds, which may provide a path to useful quantum computation in the near(er) term. Early examples of these algorithms are explored further in Chapter 4. There are many useful sources for learning about quantum computing in general, including References [132, 189]. Given theoretical speedups, a natural progression is the application of quantum computation in quantum machine learning.

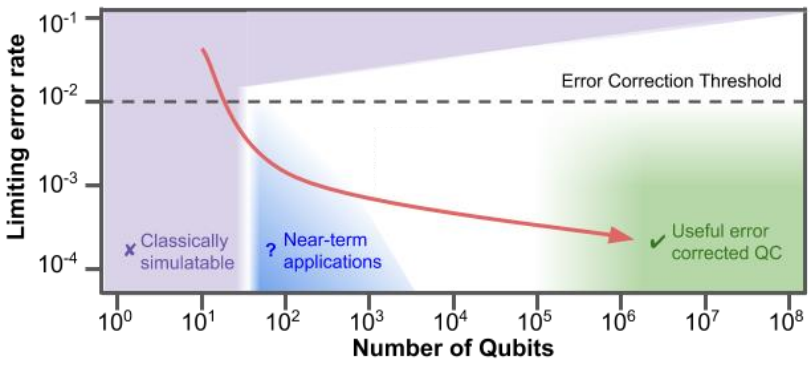


Figure 1.4: Projected development of quantum computing by Google quantum artificial intelligence lab, Figure taken from [140].

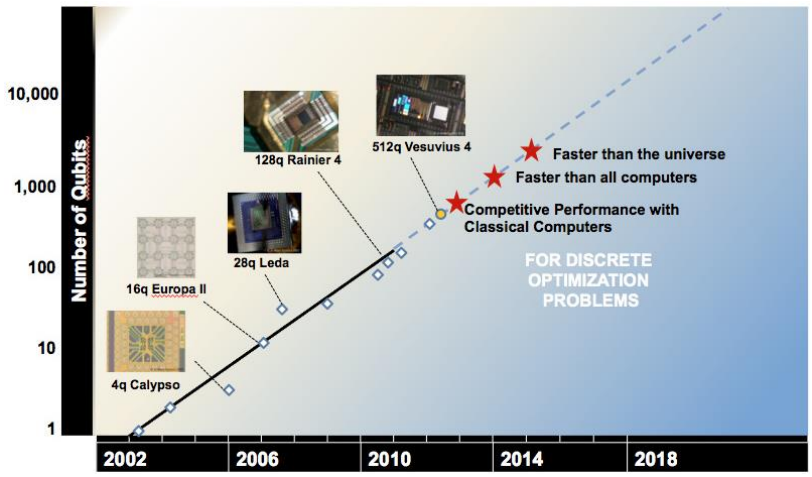


Figure 1.5: Projected development of the D-Wave quantum annealing hardware in 2010. Unfortunately, the predictions were potentially exaggerated, and a quantum computer 'faster than the universe' is not available yet, though hopes remain for 'faster than the galaxy'. Figure taken from [248]

.

1.1.3.2 Quantum machine learning

There are many examples of theoretical quantum speedup over classical machine learning algorithms: Quantum Bayesian inference [278], the quantum support vector machine [214], and quantum reinforcement learning [77], to name a few. These algorithms often exploit well known subroutines (for example HHL [155]), sometimes with quantum RAM (qRAM) [88], to provide quadratic or logarithmic speedups. They require fault tolerant (error corrected) computation with many qubits and often ignore challenging problems such as state preparation (the 'input problem'), which can require exponential time [3], and therefore may not be practical.

For problems that can be targeted by near-term quantum devices, and especially in the field of neural networks, there will be no definitive proof of superiority. Instead, analysis of performance must be done by benchmarking: The process of comparison of methods on specific problems. Evidence motivating these cases is not strong. For example, in order to motivate quantum Boltzmann machines trained via samples from a quantum annealer one has to show that quantum annealers *can* collect better representative samples of the distribution at lower cost than their classical counterparts, and also how quantum models *may* capture more useful correlations in data than purely classical models.

Unfortunately, quantifying and verifying the performance of heuristic machine learning algorithms is a hard problem. The same applies to quantum machine learning algorithms. Further, the notion of performance is an contextual amalgamation of many variables: Cost, accuracy, speed, and whatever the requirements are of the particular problem [37]. In this work these benchmarking problems are not tackled, though some comparisons are made. Here the focus is on developing useful hybrid quantum-classical frameworks that lay the groundwork for a more ambitious understanding of these early stage hybrid models.

1.2 Outline

Chapter 2 is a short background on the core concepts of quantum mechanics, including in the general formalism, the wave mechanics formalism and the abstraction to quantum computation.

In Chapter 3 we outline a quantum-assisted neural network, the quantum-assisted associative adversarial network, which exploits sampling from a quantum annealer as a subroutine in the framework. The work for this Chapter was done in collaboration with T. Vandal, T. Hogg, and E. Rieffel, and was accepted conditional on minor revisions in Quantum Machine Intelligence (Reference [283]).

After, in Chapter 4, we investigate optimisation for quantum heurisitics in the gate-model framework of quantum computation. Meta-learning, an optimisation paradigm from neural network methods, is shown to have better performance under parameter setting noise than other more standard optimisers, indicating that practical quantum heurisitics might benefit from support from neural network methods on quantum devices. The work for this Chapter was done

in collaboration with R. Stromswold, F. Wudarski, S. Hadfield, N. Tubman, and E. Rieffel, and has been accepted and is awaiting publication in Quantum Machine Intelligence (Reference [282]).

Next, in Chapters 5, we first describe the background of modelling fermionic wave functions and the development of neural network methods in the field. Chapter 6 demonstrates a state-of-the-art method, the Fermi Net [204], and various extensions, notably the application of Diffusion Monte Carlo to the Ansatz. The work for Chapter 6 was done in collaboration with N. Gao, F. Wudarski, E. Rieffel and N. Tubman, and has not begun the process of journal submission yet (Reference [281]). Finally, Chapter 7 contains concluding remarks.

Overall, this Thesis makes progress on understanding the interplay between quantum mechanics and neural networks. Initially, hybrid quantum-classical models are developed, in both quantum annealing and gate-model frameworks, and one of the problems associated with modelling quantum states with classical neural networks is tackled, achieving state-of-the-art performance. Quantum neural networks in both these guises are exciting new paradigms for solving problems. The trajectories for the improvements of the methods and hardware in both cases indicate that quantum machine learning will continue to be a fruitful research direction in the years to come.

Contributions

- Developed a hybrid quantum-classical neural network framework: the first quantum-assisted generative adversarial network and first (known) application to a complex color dataset (Chapter 3)
- Provided evidence to support neural networks as optimisers for heuristics in near-term hardware (Chapter 4)
- Improved state-of-the-art neural network methods for quantum Monte Carlo on small systems by altering the network design and extending with Diffusion Monte Carlo (Chapters 5 and 6). This is the first, to our knowledge, demonstration of Diffusion Monte Carlo with neural network methods.

CHAPTER 2

BACKGROUND

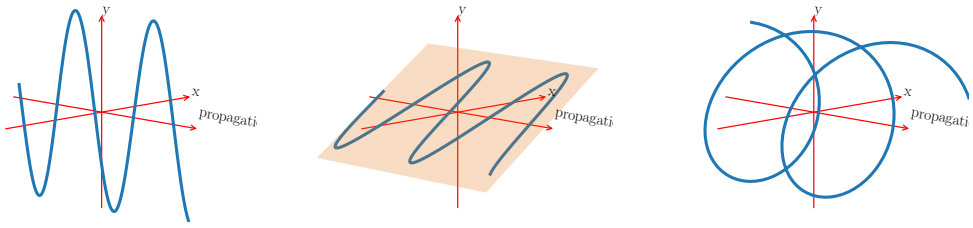
Where both reason and experience fall short,
there occurs a vacuum that can be filled by faith.
— Jostein Gardner, *Sophie's World*

This Chapter is an introduction to fundamental concepts from quantum mechanics that will be used in all sections of the Thesis. More specific background, for example methods from quantum annealing and gate-model paradigms, and literature reviews will be found before the relevant work. This description begins with the general formulation of quantum mechanics as descriptions of systems in an abstract Hilbert space via a simple experiment concerning the polarisation of photons, and goes on to relate this formulation first with quantum computation and second with the description of systems of electrons in real Euclidean space.

The aim of this Section is to provide the necessary mathematical tools to model the systems and computational frameworks used in this Thesis. Other concerns of quantum mechanics, such as the uncertainty principle, are irrelevant and as such are disregarded. For other and more complete introductions to quantum mechanics refer to References [43, 95], or sources specific to quantum computing and quantum chemistry [189, 199, 216, 249].

2.1 Quantum Mechanics

Depending on who is talking, there may be between 3 and 7 postulates of quantum mechanics [43, 72, 210]. In general, the postulates must define how to describe states of a physical system, how to define measurements on the physical system, and how to describe the evolution of the system in time (including measurement) [197]. A simple physical system, a photon, is introduced to ground abstract notions of states spaces to a physical reality.



((a)) Linearly polarised light with electric field oscillating parallel to the vertical axis.

((b)) Linearly polarised light with electric field oscillating at an angle to the vertical axis.

((c)) Circularly polarised light with a phase difference of $\pi/4$ between the orthogonal basis vectors.

Figure 2.1: Electric field of a photon illustrated in Euclidean space. Photons with electric field parallel to the polarisation of a filter can pass.

2.1.1 General formulation

2.1.1.1 Photon polarisation

A photon is a single quantum mechanical particle of light that can be modelled as a wave propagating in space. A polarisation filter is a barrier that allows the passage of photons whose electric field is oscillating parallel to its polarisation axis. Imagine a box that generates linearly polarised single photons, Figure 2.1(a). We place a polarisation filter aligned with the vertical axis some distance from the box and a detector behind the filter. The detector clicks every time a photon passes the filter and reaches the detector. At the start of the experiment the polarisation of the photons is aligned with the polarisation axis of filter and all of the photons pass through.

A second filter, whose polarisation axis parallel to the horizontal axis, is added to the experiment between the detector and the first filter. Now, no photons pass through to the detector, as the polarisation axis is orthogonal to the polarisation of the photons. Finally, a third filter is added between the two orthogonal filters. We can vary the angle of the polarisation axis of this filter relative to the vertical axis. Something strange now happens, a variable number of the photons pass through to the detector, dependent on the angle of rotation of the central polarisation filter.

States

This experiment can be described by quantum mechanics. We can describe the state of the photon polarisation with a **quantum state vector**, $|v\rangle$. $|\cdot\rangle$ is called a *ket* from Dirac notation, a particularly compact way of mathematically handling quantum mechanical objects, which will become more apparent as other concepts are introduced. The initial state of the polarisation of the photon is $|v\rangle = |\uparrow\rangle$, and, as the polarisation axis of the first filter is aligned with the vertical axis, the photon passes the filter with unit probability. Then, the second filter 'measures' the

photon, and alters the state dependent on the angle of the filter with the vertical axis. Only photons whose electric field is aligned with the filter will pass. If we decompose the polarisation of the photon as a linear combination of **basis states** aligned with ($|\nearrow\rangle$) and orthogonal to ($|\searrow\rangle$) the polarisation axis of the filter,

$$|v\rangle = \alpha(\theta)|\nearrow\rangle + \beta(\theta)|\searrow\rangle, \tag{2.1}$$

we see that there is some non-zero component of the state vector $|v\rangle$ aligned with the filter. This is a geometric decomposition of a vector into its component parts. The numbers $\alpha(\theta)$ and $\beta(\theta)$ are **amplitudes** of the polarisation of the photon in the states $|\nearrow\rangle$ and $|\searrow\rangle$, respectively. Experimentally, we find that the photon passes through the filter with **probability** $|\alpha(\theta)|^2$. In fact, as the photon must be in either state with unit probability,

$$|\alpha(\theta)|^2 + |\beta(\theta)|^2 = 1. \tag{2.2}$$

If it passes this filter, the new state of the polarisation of the photon is $|v\rangle = |\nearrow\rangle$. At each filter the polarisation of the photon is measured to determine whether or not it can pass through the filter. **Measurement** in quantum mechanics is associated with **quantum operators**. In this instance, the operator for the polarisation filter is given by the matrix

$$\hat{U} = \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix} \tag{2.3}$$

where θ is the angle with the vertical axis. This can be derived from the projectors into the basis $\{|\nearrow(\theta)\rangle, |\searrow(\theta)\rangle\}$ where explicit θ dependence is included to directly relate basis of the measurement operator $\hat{U}$ to the eigenvectors. The eigenvectors of this operator, also called the **measurement basis**, are the possible measured polarisations of an incident photon. In general, all quantum operators associated with measurements of dynamical variables must be Hermitian,

$$\hat{U} = \hat{U}^\dagger = (\hat{U}^T)^* \tag{2.4}$$

where ^{T} and * indicate the transpose and complex conjugate, respectively. The eigenvalues of a Hermitian matrix are real enforcing the condition that all physical observables be real valued. This can be derived from the conservation in time of the normalization of the wave function (**unitarity**) or from simply recognising that results of measurements must be real quantities.

In the final stage, performing the same logic as previously by decomposing the new state as a linear combination of aligned and orthogonal states, we see that there is non-zero amplitude aligned with the final polarisation filter. The final filter also measures the photon, giving the final state before the detector $|v\rangle = |\rightarrow\rangle$.

Hilbert space

For this particular experiment, we reasoned about the probability the photon passed certain filters by decomposing the state of the polarisation into a linear combination of basis states. These basis states span the entire space, that is to say *any* possible state could be described by a linear combination of these states. The system was defined as the polarisation of the photon and the space as the possible states of that system.

A system is a set of objects in a space. A space is a set of potential states (configurations) of the objects with some operations defined on the space. A vector space is a set of vectors in N dimensions that can be rescaled and added together (linear). The dimensionality of the space is defined by the number of orthonormal (orthogonal and normal) basis states, and for this system was 2. In general, the amplitudes of a quantum state vector are complex numbers, and exist in an inner product space called a Hilbert space. An inner product space is a vector space with a structure called the inner product defined for the space. The inner product $\langle \mathbf{v}_i | \mathbf{v}_j \rangle$ is a scalar quantity associated with any pair of vectors, geometrically this is interpreted as the angle between two vectors, and in a Hilbert space with orthonormal basis vectors $\{|v_1\rangle, |v_2\rangle, ..., |v_n\rangle\}$,

$$\langle \mathbf{v}_i | \mathbf{v}_j \rangle = \begin{cases} 1, & \text{if } i = j \text{ (Normal)} \\ 0, & \text{otherwise (Orthogonal)} \end{cases} \tag{2.5}$$

where $\langle \mathbf{v}_i |$ is a bra: The linear functional, a covector, of the ket $|\mathbf{v}_i\rangle$. Any bra $\langle z|$ is a linear map of all kets $|v_i\rangle$ in the Hilbert space to the complex plane $\langle z|v_i\rangle \in \mathbb{C}$. Overall, a Hilbert space is a abstract mathematical space that is complete, finite or infinite, complex-valued space with an inner product defined on the space.

In the case of the polarisation of the photon, we can use the inner product to compute the amplitude of the state aligned with the filter,

$$\langle \nearrow | v \rangle = \alpha(\theta) \overbrace{\langle \nearrow | \nearrow \rangle}^{=1} + \beta(\theta) \overbrace{\langle \nearrow | \searrow \rangle}^{=0} = \alpha(\theta). \tag{2.6}$$

This wave-like nature of quantum objects that allows states to be decomposed into linear combinations of basis states is called *superposition*.

Linearly polarised light is described in the preceding text. If we change the **phase** between the basis vectors of the quantum state, without changing the probability of measuring a basis state, for example introducing a phase difference of $\pi/4$,

$$|v\rangle = e^{i\frac{\pi}{4}} \alpha(\theta) |\nearrow\rangle + \beta(\theta) |\searrow\rangle \tag{2.7}$$

the resulting photon is circularly polarised, Figure 2.1(c). This is an example of how phase affects the properties of a system.

2.1.1.2 General systems and the Schrödinger equation

Generally, a quantum system is a collection of particles. A particle is an object with certain physical properties, which can be called dynamical variables or observables, such as spin, momentum and position, that are associated with quantum measurement operators. More concretely, an observable is any physical property or dynamical variable that can be measured by experiment [72]. Given a description of the quantum system (a quantum state), one can predict the outcome of measurements of the dynamical variable of the particles in the quantum system. The outcomes of these measurements are probabilisitic and are dependent on the amplitudes of the quantum state. The spectrum of eigenvalues of these operators are the potential measured values of the physical observable and the eigenfunctions are the corresponding states.

Of particular interest for a given system might be the energy spectrum. We can find the spectrum by solving the eigenvector equation,

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle \tag{2.8}$$

where $\hat{H}$ is the Hamiltonian, $|\psi\rangle$ is a solution (eigenvector) of the equation and E the energy (eigenvalue). This equation is called the Schrödinger equation. The time-dependent Schrödinger equation,

$$\hat{H}(t)\,|\psi(t)\rangle = -\mathrm{i}\hbar\frac{\partial\,|\psi(t)\rangle}{\partial t}, \tag{2.9}$$

describes how a quantum system evolves in time, where in both instances the Hamiltonian is determined by the system. This is the equation of motion in quantum mechanics, analogous to Newtons second law ($\mathbf{F} = m\mathbf{a}$).

Both formulations of the Schrödinger equation are *linear*: If $|\psi_0\rangle$ and $|\psi_1\rangle$ are solutions then $\alpha\,|\psi_0\rangle + \beta\,|\psi_1\rangle$ is also a solution, where α and β are complex valued scalars chosen such that the overall wave function remains normalized. It is also a *homogeneous* equation: Multiplication of a solution of the Schrödinger equation, by a scalar (rescaled) is also a solution. These are the properties of a vector space.

In this general formulation the quantum state is also referred to as the wave function. As described, this is an abstract quantity capturing the amplitude and phase of the quantum state in space and time, where the amplitude and phase are analogous to the quantities of the same name of classical waves.

Quantum computing

Although quantum mechanics describes physical systems, it is possible to abstract away from the physical concepts. This forms the basis of the framework for quantum computation. For

example, instead of discussing the polarisation of the photon we can instead talk about an abstract quantum system in a space with basis states $|0\rangle$ and $|1\rangle$. A unit of information in a classical Turing machine is called a bit, a state with value 0 or 1. The quantum analogue of the bit is the qubit, a quantum state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.10}$$

in a 2-D Hilbert space $|\psi\rangle \in \mathbb{C}^2$. As these states are unit length

$$\langle\psi|\psi\rangle = 1 \quad \text{or} \quad |\alpha|^2 + \left|\beta\right|^2 = 1. \tag{2.11}$$

They are constrained to a spherical region called the Bloch sphere. Operations on single qubit states move them around the Bloch sphere. This generalizes to systems with larger numbers of qubits to the analogue of the Bloch sphere in higher dimensions.

Two numbers are required to describe this state: α and β. A quantum system containing two qubits will require 4 numbers as any state in the space can be described as a linear combination of the basis states. In the 2 qubit case these are (in the computational basis) $\psi_\perp = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. As long as the basis states span the space, they define a valid basis. In general we may not use an orthonormal basis, but in the example outlined here the basis elements are orthonormal. The information required to describe a quantum state of multiple qubits is exponential in the number of qubits, the state $|\psi\rangle \in \mathbb{C}^{2^n}$ where n is the number of qubits. The exponential memory requirements on the description of the quantum state are the reason why classical Turing machines cannot efficiently simulate quantum systems.

In quantum computing, operations which transform qubit states are called quantum gates. Quantum gates are **unitary** operations on one or more qubits, where unitary indicates the operation maintains the normalisation of the quantum state vector. Famous examples are the Pauli-X,Y,Z gates

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.12}$$

we denote these as $\hat{X}$, $\hat{Y}$ and $\hat{Z}$ respectively for consistency with later chapters, though other typical symbolic representations are $\hat{\sigma}_x$, $\hat{\sigma}_y$ and $\hat{\sigma}_z$.

The only other gates used in this Thesis are the Hadamard gate

$$\hat{U}_{\mathrm{H}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.13}$$

Pauli rotation gates

(2.14)
$$\hat{R}_X(\theta) = \begin{pmatrix} \cos(\theta/2) & -\mathrm{i}\sin(\theta/2) \\ -\mathrm{i}\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \quad \hat{R}_Y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \quad \hat{R}_Z(\theta) = \begin{pmatrix} \mathrm{e}^{-\mathrm{i}\theta/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\theta/2} \end{pmatrix},$$

and the controlled-NOT (CNOT) gate acting on two qubits

(2.15)
$$\hat{U}_{\mathrm{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Finally, arguably essential to the potential power of quantum computation is the entanglement property of quantum states: non-classical correlations between objects. Given two qubits in the initial state $|\psi\rangle = |00\rangle$, performing a Hadamard operation on the first qubit, then a CNOT on the second conditional on the first, we create the state

(2.16)
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

The outcome of the measurement on the first qubit determines the state of the second qubit, in this instance the measurement outcomes on the qubits are perfectly correlated.

2.1.2 Wave Mechanics Formulation of Quantum Mechanics

In this Section, some aspects of the wave mechanics formulation of quantum mechanics are described, which is useful, amongst other things, for modelling systems of atoms and electrons. The descriptions of concepts in the previous section are repeated here within this new formalism.

Suppose we would like to understand an electron moving around a nucleus. The electron is a particle with properties such as position and momentum. The quantum state of a this particle can be described in a Euclidean space. In general, Euclidean space is a finite dimensional (N-D where $N < \infty$) inner product space over real numbers, for example vectors in 3-D Euclidean space can be described by $\mathbf{r} \in \mathbb{R}^3$. The inner product for a Euclidean space is usually referred to as the dot or scalar product. The wave function in position representation is a function of positions returning amplitudes. It can be represented by inner product of $|\mathbf{r}\rangle$ with the quantum state $|\psi\rangle$

(2.17)
$$\psi(x) = \langle \mathbf{r}|\psi\rangle\,.$$

$|\psi\rangle$ is the quantum state written in Dirac notation, representing a vector in a Hilbert space, $\mathbf{r}$ is point in Euclidean configuration space and $|\mathbf{r}\rangle$ is the corresponding state in Hilbert space.

This equation bridges the notions of the wave function in position representation to the general formalism described in Section 2.1.1. Both representations of the wave function are valid,

$\psi(\mathbf{r})$ and $|\psi\rangle$, and in general are used to represent the quantum state in the wave mechanics and general formulation of quantum mechanics, respectively. Of course, the wave function can be generalized to systems of n particles $\psi(\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_n, t)$ evolving in time t.

In position representation, the wave function of a single particle is a function of coordinates, $\mathbf{r} = (x, y, z)$, in 3-D Euclidean space (sometimes called the configuration space) and time, t, $\psi(\mathbf{r}, t)$. The time t and position $\mathbf{r}$ are the configuration. The probability the electron can be found at any point in space must be 1,

$$P(t) = \int \left|\psi(\mathbf{r}, t)\right|^2 d\mathbf{r} = 1, \tag{2.18}$$

where the integral is taken over all possible $\mathbf{r}$, i.e. $-\infty < x, y, z < \infty$.

There exist decompositions of the wave function into a linear sum of orthonormal functions

$$\psi(\mathbf{r}) = \sum_i \alpha_i \phi_i(\mathbf{r}) \tag{2.19}$$

where $\phi_i(\mathbf{r})$ are orthonormal

$$\int_{\mathbf{r}_1}^{\mathbf{r}_2} \psi_i(\mathbf{r})^* \psi_j(\mathbf{r}) d\mathbf{r} = \delta_{ij} \tag{2.20}$$

and the set $\Phi = \{\phi_1, \phi_2 ..., \phi_n\}$ is called the *basis*. The limits in general are over some domain $[\mathbf{r}_1, \mathbf{r}_2]$, which can be the entire continuous space $[-\infty, \infty]$. If the set of orthogonal functions $\Phi(\mathbf{r})$ can express *any* function in the space it is called complete. For the continuous space, this basis set would be infinite cardinality (size).

For a particle in a continuous 1-D Euclidean space $\psi(x, t)$ the spectrum of the position operator $\hat{x}$ is are the values x in the domain $-\infty < x < \infty$. The probabilistic nature of the wave function indicates that any single measurement of an observable is not much use by itself. Instead, an expectation value over the entire wave function must be computed

$$\left\langle \hat{O} \right\rangle = \int \psi(\mathbf{r}, t)^* \hat{O} \psi(\mathbf{r}, t) d\mathbf{r} \tag{2.21}$$

where $\hat{O}$ is the quantum operator associated with the observable and these operators are Hermitian as previously described

$$\int \psi(\mathbf{r}, t)^* \hat{O} \psi(\mathbf{r}, t) d\mathbf{r} = \int (\hat{O} \psi(\mathbf{r}, t))^* \psi(\mathbf{r}, t) d\mathbf{r}. \tag{2.22}$$

The time-independent and time-dependent Schrödinger equations of this 1-D system are written

$$\hat{H}\psi(x) = E\psi(x) \tag{2.23}$$

and

$$\hat{H}(t)\psi(x,t) = \mathrm{i}\hbar\frac{\partial\psi(x,t)}{\partial t} \tag{2.24}$$

where

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial}{\partial x^2} + V(x) \quad \text{and} \quad \hat{H}(t) = -\frac{\hbar^2}{2m}\frac{\partial}{\partial x^2} + V(x,t) \tag{2.25}$$

in this formalism.

2.1.3 Notation

Throughout the Thesis we attempt to follow some rules on notation which should make understanding equations and such easier. We state them here and largely align with commonly held practices in the literature.

Scalars are denoted by lowercase symbols, x, vectors by bold symbols $\mathbf{x}$, tensors by uppercase symbols X. Quantum operators are signified by an uppercase symbol with a hat $\hat{X}$. These rules are not maintained zealously and hopefully it is obvious when this is the case.

CHAPTER 3

DEEP LEARNING AND QUANTUM ANNEALING

Wisdom comes from experience.
Experience is often a result of lack of wisdom.
— Terry Pratchett

Generative models have the capacity to model and generate new examples from a dataset and have an increasingly diverse set of applications driven by commercial and academic interest. In this work, we present an algorithm for learning a latent variable generative model via generative adversarial learning where the canonical uniform noise input is replaced by samples from a graphical model. This graphical model is learned by a Boltzmann machine which learns low-dimensional feature representation of data extracted by the discriminator. A quantum processor can be used to sample from the model to train the Boltzmann machine. This novel hybrid quantum-classical algorithm joins a growing family of algorithms that use a quantum processor sampling subroutine in deep learning, and provides a scalable framework to test the advantages of quantum-assisted learning. For the latent space model, fully connected, symmetric bipartite and Chimera graph topologies are compared on a reduced stochastically binarized MNIST dataset, for both classical and quantum sampling methods. The quantum-assisted associative adversarial network successfully learns a generative model of the MNIST dataset for all topologies. Evaluated using the Fréchet inception distance and inception score, the quantum and classical versions of the algorithm are found to have equivalent performance for learning an implicit generative model of the MNIST dataset. Classical sampling is used to demonstrate the algorithm on the LSUN bedrooms dataset, indicating scalability to larger and color datasets. Though the quantum processor used here is a quantum annealer the algorithm is general enough such that any quantum processor, such as gate-model quantum computers, may be substituted as a sampler.

3.1 Introduction

The ability to efficiently and accurately model a dataset, even without full knowledge of why a model is the way it is, is a valuable tool for understanding complex systems. Machine Learning (ML), the field of data analysis algorithms that create models of data, is experiencing a renaissance due to the availability of data, increased computational resources and algorithm innovations, notably in deep neural networks [239, 240]. Of particular interest are unsupervised algorithms that train generative models. These models are useful because they can be used to generate new examples representative of a dataset.

A GAN is an algorithm which trains a latent variable generative model with a range of applications including image or signal synthesis, classification and upscaling. The algorithm has been demonstrated in a range of architectures, now well over 300 types and applications, from the GAN zoo [115, 146, 209]. Two problems in GAN learning are non-convergence, oscillating and unstable parameters in the model, and mode collapse, where the generator only provides a small variety of possible samples. These problems have been addressed previously in existing work including energy based GANs [296] and the Wasserstein GAN [19, 100]. Another proposed solution involves replacing the canonical uniform noise prior of a GAN with a prior distribution modelling low-dimensional feature representation of the dataset. Using this informed prior may alleviate the learning task of the generative network, decrease mode-collapse and encourage convergence [18].

This feature distribution is a rich and low-dimensional representation of the dataset extracted by the discriminator in a GAN. A generative probabilistic graphical model can learn this feature distribution. However, given the intractability of calculating the exact distribution of the model, classical techniques often use approximate methods for sampling from restricted topologies, such as contrastive divergence, to train and sample from these models. Novel means for sampling efficiently and accurately from less restricted topologies could further broaden the application and increase the effectiveness of these already powerful approaches.

Quantum computing can provide proven advantages for some sampling tasks; for example, it is known that under reasonable complexity theory assumptions, quantum computers can sample more efficiently from certain classical distributions [44, 156]. It is an open question in quantum computing as to the extent to which quantum computers provide a more efficient means of sampling from other, more practically useful, distributions. On gate model quantum computers, quantum sampling for machine learning has been explored by a number of groups for arbitrary distributions [31, 81, 230] and Boltzmann distributions [236, 262, 298]. In a quantum annealing framework, there is literature on sampling from Boltzmann distributions [12, 33]. Here, we extend this exploration to the use of quantum processors for Boltzmann sampling to the powerful framework of adversarial learning. Specifically, we model the latent space with a Boltzmann machine trained via samples from a quantum annealer, though any classical or quantum system for sampling from a parameterized distribution could be used. Our experiments used the D-Wave

2000Q quantum annealer, but the work is relevant for near-term quantum processors in general.

Quantum annealing has been shown to sample from a Boltzmann-like distribution on near-term hardware [12, 33] effectively enough to train Boltzmann machines and to exhibit learning. In the future, quantum annealing may decrease the cost of this training by decreasing the computation time [37], energy usage [65], or improve performance as quantum models [124] may better represent some datasets.

Here, we demonstrate the Quantum-Assisted Associative Adversarial Network (QAAAN) algorithm, Figure 3.1, a hybrid quantum-assisted GAN in which a BM trains, using samples from a quantum annealer, a model of a low-dimensional feature distribution of the dataset as the prior to a generator. The model learned by the algorithm is a latent variable implicit generative model $p(x \mid z)$ and an informed prior $p(z)$, where z are latent variables and x are data space variables. The prior will contain useful information about the features of the data distribution and this information will not need to be learned by the generator. Put another way, the prior will be a model of the feature distribution containing the latent variable modes of the dataset.

Contributions

The core contribution of this work is the development of a scalable quantum-assisted GAN which trains an implicit latent variable generative model. This algorithm fulfills the criteria for inclusion of near-term quantum hardware in deep learning frameworks that can learn continuous variable datasets: Resistant to noise, small number of variables, in a hybrid architecture. Additionally in this work we explore different topologies for the latent space model. The contributions of this work are

Contributions

- compared different topologies to appropriately choose a graphical model, restricted by the connectivity of the quantum hardware, to integrate with the deep learning framework
- designed a framework for using sampling from a quantum processor in generative adversarial networks, which may lead to architectures that encourage convergence and decrease mode collapse,
- and lastly designed a hybrid quantum-classical framework which successfully tackles the problems of integrating a quantum processor into scalable deep learning frameworks, whilst exploiting the strengths of classical elements (handling a large number of continuous variables) and the quantum processor (sampling hard distributions).

These contributions extend previous work for integrating quantum processors into deep learning frameworks [29, 30, 129, 200]. The Quantum-Assisted Helmholtz Machine (QAHM)

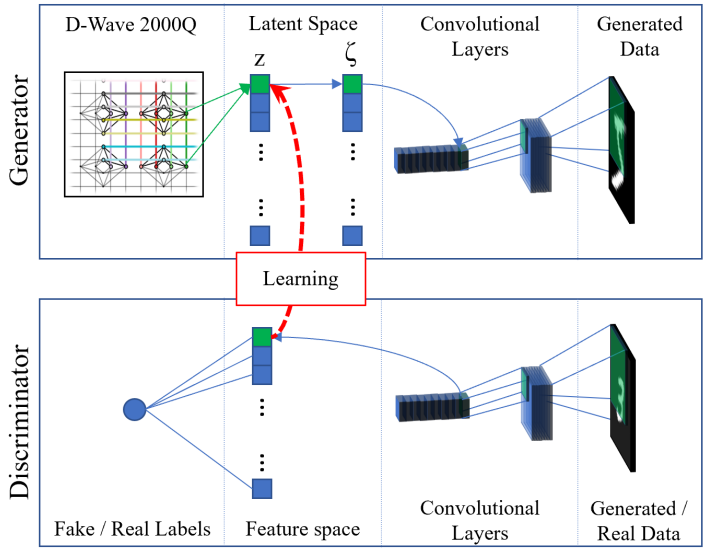


Figure 3.1: The inputs to the generator network are samples from a Boltzmann distribution. A BM trains a model of the feature space in the generator network, indicated by the Learning. Samples from the quantum annealer, the D-Wave 2000Q, are used in the training process for the BM, and replace the canonical uniform noise input to the generator network. These discrete variables $\mathbf{z}$ are reparametrised to continuous variables ζ before being processed by transposed convolutional layers. Generated and real data are passed into the convolutional layers of the discriminator which extracts a low-dimensional representation of the data. The BM learns a model of this representation. An example flow of information through the network is highlighted in green. In the classical version of this algorithm, MCMC sampling is used to sample from the discrete latent space, otherwise the architectures are identical.

[29, 200] was one of the first attempts to integrate quantum models into the latent space of deep learning architectures. However, the QAHM is based on the *wake-sleep* algorithm, which faces two challenges: the loss function is not well defined and the gradients do not propagate between the inference and generative networks. The quantum variational autoencoder [129] tackles both of these challenges. We introduce another algorithm which also handles these challenges and has additional advantages, including high fidelity images, for which GANs are generally known to perform well. Finally, after completing the work, a preprint was posted that covers similar ideas on quantum-classical associative adversarial networks that was performed independently from ours. In this work, the authors investigate a quantum-classical associative model and sample the latent space with quantum Monte Carlo [15].

Outline

First, there is a short background section, specifically GANs, quantum annealing, and Boltzmann machines. In Section 3.3 an algorithm is developed to learn a latent variable generative model

using samples from a quantum processor to replace the canonical uniform noise input. We explore different models, specifically complete, symmetric bipartite and Chimera topologies, tested on a reduced stochastically binarized version of MNIST, for use in the latent space. In Section 3.4 the results are detailed, including application of the QAAAN and a classical version of the algorithm to the MNIST dataset. The architectures are evaluated using the Inception Score and the Frechét Inception Distance. The algorithm is also implemented on the LSUN bedrooms dataset using classical sampling methods, demonstrating the scalability.

3.2 Background

3.2.1 Generative Adversarial Networks

Implicit generative models are those which specify a stochastic procedure with which to generate data. In the case of a GAN, the generative network maps latent variables $\mathbf{z}$ to images which are likely under the real data distribution, for example $\mathbf{x} = G(\mathbf{z})$, G is the function represented by a neural network, $\mathbf{x}$ is the resulting image with $\mathbf{z} \sim q(\mathbf{z})$, and $q(\mathbf{z})$ is typically the uniform distribution between 0 and 1, $\mathscr{U}[0,1]$.

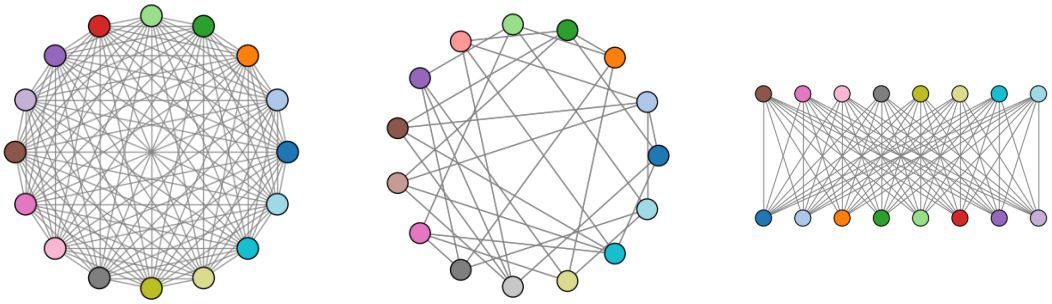
Training a GAN can be formulated as a minimax game where the discriminator attempts to maximise the cross-entropy of a classifier that the generator is trying to minimise. The cost function of this minimax game is

$$\begin{aligned} V(D,G) = {} & \mathbb{E}_{x\sim p(x)}[\log(D(x))] \\ & + \mathbb{E}_{z\sim q(z)}[\log(1-D(G(z)))]. \end{aligned} \tag{3.1}$$

$\mathbb{E}_{x\sim p(x)}$ is the expectation over the distribution of the dataset, $\mathbb{E}_{z\sim q(z)}$ is the expectation over the latent variable distribution and D and G are functions instantiated by a discriminative and generative neural network, respectively, and we are trying to find $\min_G \max_D V(D,G)$. The model learned is a latent variable generative model $P_{model}(x \mid z)$.

The first term in Equation (3.1) is the log-probability of the discriminator predicting that the real data is genuine and the second the log-probability of it predicting that the generated data is fake. In practice, ML engineers will instead use a heuristic maximising the likelihood that the generator network produces data that trick the discriminator instead of minimising the probability that the discriminator label them as real. This has the effect of stronger gradients earlier in training [93].

GANs are lauded for many reasons: The algorithm is unsupervised; the adversarial training does not require direct replication of the real dataset resulting in samples that are sharp [268]; and it is possible to perform the weight updates through efficient backpropagation and stochastic gradient descent. There are also several known disadvantages. Primarily, the learned distribution is implicit. It is not straightforward to compute the distribution of the training set [178] unlike explicit, or prescribed, generative models which provide a parametric specification of



((a)) ((b)) ((c))

Figure 3.2: (a) Complete (b) Chimera (c) symmetric bipartite graphical models. These graphical models are embedded into the hardware and the nodes in these graphs are not necessarily representative of the embeddings.

the distribution specifying a log-likelihood $\log P(\mathbf{x})$ that some observed variable $\mathbf{x}$ is from that distribution. This means that simple GAN implementations are limited to generation.

Further, as outlined in the introduction, the training is prone to non-convergence [26], and mode collapse [251]. This stability of GAN training is an issue and there are many hacks to encourage convergence, discourage mode-collapse and increase sample diversity including using spherical input space [276], adding noise to the real and generated samples [19] and minibatch discrimination [225]. We hypothesise that using an informed prior will decrease mode-collapse and encourage convergence.

3.2.2 Boltzmann Machines & Quantum Annealing

A BM is a energy-based graphical model composed of stochastic nodes, with weighted connections between and biases applied to the nodes. The energy of the network corresponds to the energy function applied to the state of the system. BMs represent multimodal and intractable distributions [143], and the internal representation of the BM, the weights and biases, can learn a generative model of a distribution [5].

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with cardinality N describing a Boltzmann machine with model parameters $\boldsymbol{\lambda} = \{\boldsymbol{\omega}, \boldsymbol{b}\}$ over logical variables $\mathcal{V} = \{z_1, z_2, ...z_N\}$ connected by edges $\mathcal{E}$ has energy

$$E_{\boldsymbol{\lambda}}(\mathbf{z}) = -\sum_{z_i \in \mathcal{V}} b_i z_i - \sum_{(z_i, z_j) \in \mathcal{E}} \omega_{ij} z_i z_j \tag{3.2}$$

where weight ω_{ij} is assigned to the edge connecting variables z_i and z_j, bias b_i is assigned to variable z_i and possible states of the variables are $z_i \in \{-1, 1\}$ corresponding to 'off' and 'on', respectively. We refer to this graph as the logical graph. The distribution of the states $\mathbf{z}$ is

$$P(\mathbf{z}) = \frac{e^{-\beta E_{\boldsymbol{\lambda}}(\mathbf{z})}}{Z} \tag{3.3}$$

with β a parameter recognized by physicists as the inverse temperature in the function defining the Boltzmann distribution and $Z = \sum_{\mathbf{z}} e^{-\beta E_\lambda(\mathbf{z})}$ the partition function, where the sum is over all possible states.

BM training requires sampling from the distribution represented by Equation (3.3). For fully-connected variants it is an intractable problem to calculate the probability of the state occurring exactly [134] and is computationally expensive to approximate. Exact inference of complete graph BMs is generally intractable and approximate methods including Gibbs sampling are slow. Generally, applications will use deep stacked Restricted Boltzmann Machine (RBM) architectures, which can be efficiently trained with approximate methods, notably contrastive divergence. Contrastive divergence and Gibbs sampling are examples of Markov Chain Monte Carlo (MCMC) methods.

A Markov chain describes a sequence of stochastic states where the probability of any state in the chain occuring is only dependent on the previous state. A Monte Carlo method is one which uses repeated sampling to make numerical approximations. For example, Gibbs sampling involves taking some random initial state of nodes, stochastically updating the state of a random node z_n given the current states of other nodes $P(z_n = 1|\mathcal{V} \setminus z_n) = S(2\beta(\sum_{z_i \in \mathcal{V} \setminus z_n} \omega_{ni} z_i + b_n))$, where

$$S(x) = \frac{1}{1+e^{-x}}. \tag{3.4}$$

Repeating this procedure many times evolves the Markov chain to some equilibrium where samples of states are approximately from the Boltzmann distribution, Equation (3.3).

An RBM is a symmetric bipartite BM, shown in Figure 3.2(c). It is possible to efficiently learn the distribution of some input data spaces through approximate methods, e.g. contrastive divergence [54]. Stacked RBMs form a Deep Belief Network (DBN) and can be greedily trained to learn the generative model of datasets with higher-level features with applications in a wide range of fields from image recognition to finance [70]. Training these types of models requires sampling from the Boltzmann distribution.

Quantum Annealing (QA) has been proposed as a method for sampling from complex Boltzmann-like distributions. It is an optimisation algorithm exploiting quantum phenomena to find the ground state of a cost function. QA has been demonstrated for a range of optimisation problems [39], however, defining and detecting speedup, especially in small and noisy hardware implementations is challenging [126, 219].

QA has been proposed and in some cases demonstrated as a sampling subroutine in ML algorithms: A quantum Boltzmann machine [12]; training a Quantum Variational Autoencoder (QVAE) [129]; a quantum-assisted Helmholtz machine [29]; deep belief nets of stacked RBMs [6].

In order to achieve this, the framework outlined in Equation (3.2) can be mapped to an Ising

model for a quantum system represented by the Hamiltonian

$$\hat{H}_{\boldsymbol{\lambda}} = -\sum_{\hat{\sigma}_i^z \in \mathcal{V}} h_i \hat{\sigma}_i^z - \sum_{(\hat{\sigma}_i^z, \hat{\sigma}_j^z) \in \mathcal{E}} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z. \tag{3.5}$$

where now variables z have been replaced by the Pauli-z operators, $\hat{\sigma}_i$, which return eigenvalues in the set $\{-1, 1\}$ when applied to the state of variable z_i, physically corresponding to spin-up and spin-down, respectively. Parameters b_i and ω_{ij} are replaced with the Ising model parameters h_i and J_{ij} which are conceptually equivalent. In the hardware, these parameters are referred to as the flux bias and the coupling strength, respectively.

The full Hamiltonian describing the dynamics of the D-Wave 2000Q, equivalent to the time-dependent transverse field Ising model, is

$$\hat{H}(t) = A(t)\hat{H}_{\perp} + B(t)\hat{H}_{\boldsymbol{\lambda}}. \tag{3.6}$$

The transverse field term $H_{\perp}$ is

$$\hat{H}_{\perp} = \sum_{\hat{\sigma}_i^x \in \mathcal{V}} \hat{\sigma}_i^x. \tag{3.7}$$

$\hat{\sigma}^x$ are the Pauli-x operators in the Hilbert space $\mathbb{C}^{2^N}$. $A(t)$ and $B(t)$ are monotonic functions defined by the total annealing time $t_{\max}$ [39]. Generally, at the start of an anneal, $A(0) \approx 1$ and $B(0) \approx 0$. $A(t)$ decreases and $B(t)$ increases monotonically with t until, at the end of the anneal, $A(t_{\max}) \approx 0$ and $B(t_{\max}) \approx 1$. When $B(t) > 0$, the Hamiltonian contains terms that are not possible in the classical Ising model, that is those that are normalised linear combinations of classical states.

This Hamiltonian was embedded in the D-Wave 2000Q, a system with 2048 qubits, each with degree 6, i.e. other than qubits on the edge of the graph each qubit is connected to 6 other qubits. Embedding is the process of mapping the logical graph, represented by Equation (3.5), to hardware. If the logical graph has degree > 6 or a structure that is not native to the hardware, the logical graph can still be embedded in the hardware via a 1-many mapping, that means one variable z_i is represented by more than one qubit. These qubits are arranged in a 'chain'(this term is used even when the set of qubits forms a small tree). A chain is formed by setting the coupling strength J_{ij} between these qubits to a strong value to encourage them to take a single value by the end, but not so strong that it overwhelms the J_{ij} and h_i in the original problem Hamiltonian or has a detrimental effect on the dynamics. There is a sweet spot for this value. In our case, we used the maximum value available on the D-Wave 2000Q, namely -1. At the end of the anneal, to determine the value of a logical variable expressed as a qubit chain in the hardware a majority vote is performed: The logical variable takes the value corresponding to the state of the majority of qubits. If there is no majority a coin is flipped to determine the value of the logical variable.

Each state found after an anneal comes from a distribution, though it is not clear what distribution the quantum annealer is sampling from. For example, in some cases the distribution is

Algorithm 1 Quantum-assisted associative adversarial network training.

1: **for** iterations **do**
% Train the discriminator
2: Sample n Boltzmann distribution samples from $\rho \rightarrow \boldsymbol{\phi} = \{\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_n\}$ using quantum annealer
3: Sample n training data examples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ from the dataset
4: Generate $\mathbf{X}^D = G(\boldsymbol{\phi})$ where $\mathbf{X}^D = \{\mathbf{x}_1^D, \mathbf{x}_2^D, ..., \mathbf{x}_n^D\}$
5: $\theta_D \leftarrow \theta_D - \nabla_{\theta_D} \sum_i^n \left(\log D(\mathbf{x}_i) + \log\left(D(\mathbf{x}_i^D)\right)\right)$
% Train the associative model
6: Generate $\boldsymbol{\phi}_f = D(\mathbf{X})$ where $\phi_f = \{\mathbf{z}_1^D, \mathbf{z}_2^D, ..., \mathbf{z}_n^D\}$
7: Update weights of BM via SGD with $\boldsymbol{\phi}_f$ and $\boldsymbol{\phi}$ via Equations (3.11) and (3.12)
% Train the generator
8: $\theta_G \leftarrow \theta_G - \nabla_{\theta_G} \sum_i^n \left(\log\left(D(\mathbf{x}_i^D)\right)\right)$
9: **end for**
10: **return** Network $G(z; \theta_G)$

Figure 3.3: QAAAN training algorithm. ρ represents the distribution sampled by the quantum annealer, therefore $\rho \rightarrow \phi$ represents sampling a set of vectors $\mathbf{z}_i$ from distribution ρ. The training samples, $\mathbf{X}$, are sampled from the datasets MNIST or LSUN bedrooms. Steps 5 and 8 are typical of GAN implementation, $G(\cdot)$ and $D(\cdot)$ are the functions representing the generator and discriminator networks, respectively. For clarity, we have omitted implementation details arising from the embedding a logical graph into the quantum annealer. Further details on mapping to the logical space for samples from the quantum annealer can be found in Section 3.3.

hypothesised to follow a quantum Boltzmann distribution up to the 'freeze-out region' - where for some value of t the dynamics of the system slow down and diverge from the model, Equation (3.8) [11]. If the freeze-out region is narrow then the distribution can be modelled as the classical distribution of problem Hamiltonian, $\hat{H}_{\boldsymbol{\lambda}}$, at a higher unknown effective temperature β^*,

$$\rho_{t_{\max}} = \frac{e^{-\beta^* \hat{H}_{\boldsymbol{\lambda}}}}{Z} \tag{3.8}$$

where $Z = \mathrm{Tr}\left[e^{-\beta^* \hat{H}_{\boldsymbol{\lambda}}}\right]$ and we have performed matrix exponentiation. $\rho_{t_{\max}}$ is the model for the distribution. In the case where the dynamics of the distribution do not slowdown and $\hat{H}(t_{\max}) = \hat{H}_{\lambda}$ the Hamiltonian contains no off-diagonal terms and Equation (3.8) is equivalent to the classical Boltzmann distribution, Equation (3.3), at some temperature. β is a dimensionless parameter which depends on the temperature of the system, the energy scale of the superconducting flux qubits and open system quantum dynamics. However, it is an open question as to when the freeze-out hypothesis holds [8, 161].

Other implementations of training graphical models have accounted for this instance dependent effective temperature [32], in this work to get around the problem of using the unknown effective temperature for training a probabilistic graphical model, we use a gray-box model ap-

proach proposed in [33]. In this approach, full knowledge of the effective parameters, dependent on β, are not needed to perform the weight updates as long as the projection of the gradient is positive in the direction of the true gradient. The gray-box approach ties the model generated to the specific device used to train the model, though is robust to noise and is *not* required to estimate β [213]. The computation of the gradients is described further in Section 3.3.1. We find that under this approach performance remains good enough for deep learning applications.

Though we do not have full knowledge of the distribution the quantum annealer samples from, we have modelled it as a classical Boltzmann distribution at an unknown temperature. This allows us to train models without the having to estimate the temperature of the system, providing a simple approach to integrating probabilistic graphical models into deep learning frameworks.

3.3 Quantum-assisted associative adversarial network

In this section, the QAAAN algorithm is outlined, including a novel way to learn the feature distribution generated by the discriminator network via a BM using sampling from a quantum annealer. The QAAAN architecture is similar to the classical Associative Adversarial Network proposed in Ref [18], as such the minimax game played by the QAAAN is

$$\begin{aligned} V(D,G,\rho) = &\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] \\ &+ \mathbb{E}_{z\sim\rho(z)}[\log(1-D(G(z)))] \\ &+ \mathbb{E}_{f\sim\rho_f(f)}[\log\rho], \end{aligned} \tag{3.9}$$

where the aim is now to find $\min_{\mathscr{G}}\max_{\rho}\max_{\mathscr{D}} V(D,G,\rho)$, with equivalent terms to Equation (3.1) plus an additional term to describe the optimisation of the model ρ, Equation (3.8). This term conceptually represents the probability that samples generated by the model ρ are from the feature distribution ρ_f. ρ_f is the feature distribution extracted from the interim layer of the discriminator. This distribution is assumed to be Boltzmann, a common technique for modelling a complex distribution.

The algorithm used for training ρ, a probabilistic graphical model, is a BM. Sampling from the quantum annealer, the D-Wave 2000Q, replaces a classical sampling subroutine in the BM. ρ is used in the latent space of the generator, Figure 3.1, and samples from this model, also generated by the quantum annealer, replace the canonical uniform noise input to the generator network. Samples from ρ are restricted to discrete values, as the measured values of qubits are $\mathbf{z}\in\{-1,+1\}$. These discrete variables $\mathbf{z}$ are reparametrised to continuous variables $\boldsymbol{\zeta}$ before being processed by the layers of the generator network, producing 'generated' data. Generated and real data are then passed into the layers of the discriminator which extracts the low-dimensional feature distribution ρ_f. This is akin to a variational autoencoder, where an approximate posterior maps the evidence distribution to latent variables which capture features of the distribution [71]. The algorithm for training the complete network is detailed in Algorithm 1.

Below, we outline the details of the BM training in the latent space, reparametrisation of discrete variables, and the networks used in this investigation. Additionally, we detail an experiment to distinguish the performance of three different topologies of probabilistic graphical models to be used in the latent space.

3.3.1 Latent space

As in Figure 3.1, samples from a intermediate layer of the discriminator network are used to train a model for the latent space of the generator network. Here, a BM trains this model. The cost function of this BM is the quantum relative entropy

$$S(\rho||\rho_f) = \mathrm{Tr}\big[\rho \ln \rho\big] - \mathrm{Tr}\big[\rho \ln \rho_f\big] \tag{3.10}$$

equivalent to the classical Kullback-Leibler divergence when all off-diagonal elements of ρ and ρ_f are zero. This measure quantifies the divergence of distribution ρ from ρ_f where ρ_f is the target feature distribution of features extracted by the discriminator network and ρ is the model trained by the BM, from Equation (3.9). Though the distributions used here are modelled classically, this framework can be extended to quantum models using the quantum relative entropy. Given this it can be shown that the updates to the weights and biases of the model are

$$\Delta J_{ij} = \eta\beta[\langle z_i z_j\rangle_{\rho_f} - \langle z_i z_j\rangle_{\rho}] \tag{3.11}$$

$$\Delta h_i = \eta\beta[\langle z_i\rangle_{\rho_f} - \langle z_i\rangle_{\rho}]. \tag{3.12}$$

η is the learning rate, β is an unknown parameter, and $\langle z\rangle_\rho$ is the expectation value of z in distribution ρ. Under the gray box model, discussed in Section 3.2.2, we set $\beta = 1$ and tune the learning rate without knowledge of the true value of β. z are the logical variables of the graphical model and the expectation values $\langle z\rangle_\rho$ are estimated by averaging 1000 samples from the quantum annealer. The quantum relative entropy is minimised by stochastic gradient descent.

3.3.2 Topologies

We explored three different topologies of probabilistic graphical models, complete, symmetric bipartite and Chimera, for the latent space. Their performance on learning a model of a reduced stochastically binarized version of MNIST, Figure 3.4, was compared, in both sampling via quantum annealing and classical sampling cases. The complete topology is self-explanatory, Figure 3.2(a), restricted refers to a symmetric bipartite graph, Figure 3.2(c), and the sparse is the graph native to the D-Wave 2000Q, or Chimera graph, where the connectivity of the model is determined by the available connections on the hardware, Figure 6.14(c).

The models were trained by minimising the quantum relative entropy, Equation (3.10), and evaluated with the L_1-norm,

$$L_1\text{-norm} = \sum_{z_i, z_j \in \mathcal{V}} \Big| \langle z_i z_j\rangle_{\rho_f} - \langle z_i z_j\rangle_{\rho} \Big|. \tag{3.13}$$

The algorithm did not include temperature estimation, or methods to adjust intra-chain coupling strengths for the embedding, as in [32] and [33], respectively. The method used here makes a comparison between the different topologies, though for best performance one would want to account for the embedding and adjust algorithm parameters, such as the learning rate, to each topology.

In addition to these requirements, there are several non-functioning, 'dead', qubits and couplers in the hardware. These qubits or couplers were removed in all embeddings, which had a negligible effect on the final performance. The complete topology embedding was found using a heuristic embedder [48]. A better choice would be a deterministic embedder, resulting in shorter chain lengths, though when adjusting for the dead qubits the symmetries are broken and the embedded graph chain length increases to be comparable to that returned by the heuristic embedder. The restricted topology was implemented using the method detailed by Adachi and Henderson [6]. At a high level, qubits corresponding to visible nodes are mapped to vertical chains and qubits corresponding to hidden nodes are mapped to horizontal chains . The Chimera topology was implemented on a 2x2 grid of unit cells, avoiding dead qubits. Learning was run over 5 different embeddings for each topology and the results averaged. For topologies requiring chains of qubits, the couplers in the chains were set to -1.

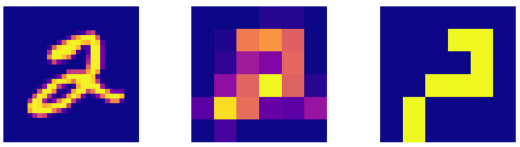


Figure 3.4: Left to right: 28x28 continuous, 6x6 continuous, 6x6 stochastically binarized example from the MNIST dataset.

3.3.3 Reparametrisation

Samples from the latent space come from a discrete space. These variables are reparametrised to a continuous space, using standard techniques. There are many potential choices for reparametrisation functions and a simple example case is outlined below. We chose a probability density function pdf(x) which rises exponentially and can be scaled by parameter α:

$$\mathrm{p}(x) = \frac{\alpha \exp(-\alpha(1-x))}{1-\exp(-2\alpha)}. \tag{3.14}$$

The cumulative distribution function of this probability density function is

$$F(r) = \begin{cases} 1 & r > 1 \\ \int_{-1}^{r} p(x)\,dx & -1 < r \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{3.15}$$

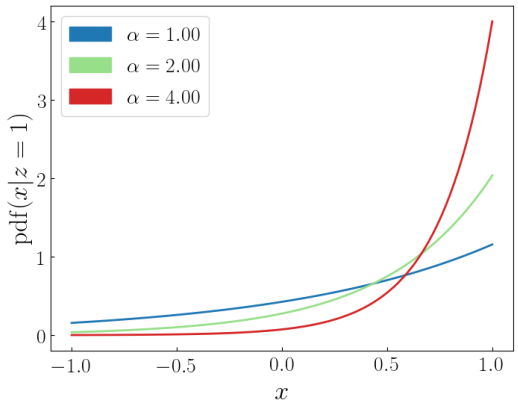


Figure 3.5: The probability density function, $p(x)$, for different values of α. In this investigation $\alpha = 4$ was used, to distinguish strongly from the uniform noise case.

and

$$\int_{-1}^{r} p(x)\,dx = \frac{\exp(-\alpha(1-r)) - \exp(-2\alpha)}{1-\exp(-2\alpha)} \tag{3.16}$$

Discrete samples can be reparametrised by sampling $F(r)$ from $\mathcal{U}(0,1]$ solving Equation (3.16) for r. In the method implemented in this work, the continuous variables ζ_i, Figure 3.1, are -1 in the case when $z_i = -1$ and sampled from pdf(x), Equation (3.14), when $z_i = 1$. Then the continuous variables are input to the generator of the network.

The value of α was set to 4. The uncertainty in the evaluation of performance was not accurate enough to determine meaningful differences in the setting of α. We chose α to intuitively set ζ_i close to the discrete variables z_i. Certainly, further investigation into the type of reparameterization, e.g. different functions and methods developed in other works [12], will be needed to determine best practises in the field for these types of models. Here, we restrict the research to the development of a new model.

3.3.4 Networks

The generator network consists of dense and transpose convolutional, stride 2 kernel size 4, layers with batch normalisation and ReLU activations. A ReLU activation is a standard activation function used to avoid vanishing gradients, where the derivative is 0 when the input is less than 0 and 1 otherwise. The output layer is implemented with a tanh activation. These components are standard deep learning techniques found in textbooks, for example [92].

The discriminator network consists of dense, convolutional layers, stride 2 kernel size 4, LeakyReLU activations. The dense layer corresponding to the feature distribution was chosen to have tanh activations in order that outputs could map to the BM. The hidden layer representing ρ_f was the fourth layer of the discriminator network with 100 nodes. When sampling the training

data for the BM from the discriminator, the variables given values from the set $\{-1, 1\}$ as in the Ising model, dependent on the activation of the node being greater or less than the threshold, set at zero, respectively.

We found the stability of the network was dependent on the structure of the model and followed best practises [19, 93] to guide the development. Though this model worked well there is scope to understand if there are alternate best practises for models incorporating quantum devices into the latent space.

The networks were trained with an Adam optimiser with learning rate 0.0002. The learning rate was set by performing two sweeps. The first over the set {0.01, 0.001, 0.0001, 0.00001} and the second in the region around the best performing (0.001). The best performing was determined by evaluating the Inception Score and the Fréchet Inception Distance (FID), described further in Section 3.4, after 100 iterations. High learning rates (0.01) were unstable and the training diverged, whereas low learning rates (0.00001) the training was stable but slow. The labels were smoothed with noise.

For the sparse graph latent space used in learning the MNIST dataset in Section 3.4, the BM was embedded in the D-Wave hardware using a heuristic embedder. As there is a 1-1 mapping for the sparse graph it was expressed in hardware using 100 qubits. An annealing schedule of $1\mu s$ and a learning rate of 0.0002 were used. The classical architecture that was compared with the QAAAN was identical other than replacing sampling via quantum annealing with MCMC sampling techniques.

We used D-Wave Systems inc. provided Python API to interact with the device. Qubits, biases and weights can be assigned through this API, which can be used to implement details such as the embedding. Other experimental hyperparameters (number of samples/annealing schedule) can be set with high-level functionality.

3.4 Results & Discussion

For this work we performed several experiments. First, we compared three topologies of graphical models, trained using both classical and quantum annealing sampling methods. They were evaluated for performance by measuring the L_1-norm over the course of the learning a reduced stochastically binarzied version of the MNIST dataset, Figure 3.4. Second, the QAAAN and the classical associative adversarial network described in Section 3.3 were both used to generate new examples of the MNIST dataset. Their performance was evaluated used the inception score and the FID. Finally, the classical associative adversarial network was used to generate new examples of the LSUN bedrooms dataset.

In the experiment comparing topologies, as expected, the BM trains a better model faster with higher connectivity, Figure 3.7. When trained via sampling with the quantum annealer the picture is less intuitive, Figure 3.6. All topologies learned a model to the same accuracy, at similar

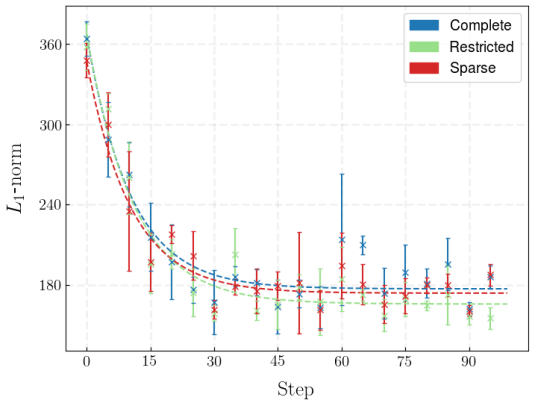


Figure 3.6: Comparison of the convergence of different graphical topologies trained using samples from a quantum annealers on a reduced stochastically binarized MNIST dataset. The learning rate used was 0.03. This learning rate produced the fastest learning with no loss in performance of the final model. The learning was run 5 times over different embeddings and the results averaged. The error bars describe the variance over these curves.

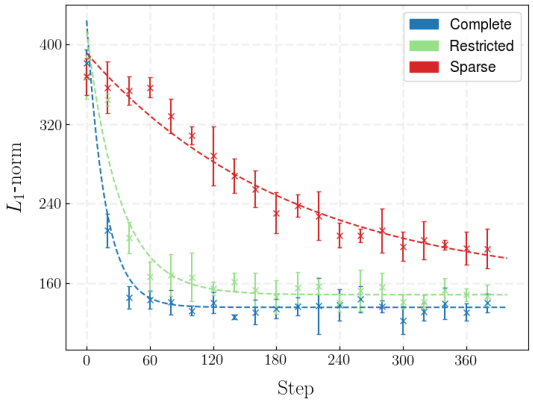


Figure 3.7: Comparison of different graphical topologies trained using MCMC sampling on a reduced stochastically binarized MNIST dataset. The learning rate used was 0.001. This learning rate was chosen such that the training was stable for each topology, we found that the error diverged for certain topologies at other learning rates. The learning was run 5 times and the results averaged. The error bars decribe the variance over these curves.

rates. This indicates that there is a noise floor preventing the learning of a better model in the more complex graphical topologies. For the purposes of this investigation the performance of the sparse graph was demonstrated to be enough to learn an informed prior for use in the QAAAN algorithm.

Given the results of the first experiment, the classical associative adversarial network and the quantum-assisted algorithm were evaluated with a sparse topology latent space. The gen-

erated images are shown for both classical and quantum versions in Figures 3.8(a) and 3.8(b), respectively.

We evaluated classical and quantum-assisted versions of the associative adversarial network with sparse latent spaces via two metrics, the inception score and the FID. Both metrics required an inception network, a network trained to classify images from the MNIST dataset, which was trained to an accuracy of ~ 95%. The Inception Score, Equation (3.17), attempts to quantify realism of images generated by a model. For a given image, $p(y|x)$ should be dominated by one value of y, indicating a high probability that an image is representative of a class. Secondly, over the whole set there should be a uniform distribution of classes, indicating diversity of the distribution. This is expressed

$$\mathrm{IS} = \exp\left(\mathbb{E}_{x\sim\rho_{\mathrm{D}}}\, \mathrm{D}_{\mathrm{KL}}(p(y|x)||p(y))\right). \tag{3.17}$$

The first criterion is satisfied by requiring that image-wise class distributions should have low entropy. The second criterion implies that the entropy of the overall distribution should be high. The method is to calculate the KL distance between these two distributions: A high value indicates that both the $p(y|x)$ is distributed over one class and $p(y)$ is distributed over many classes. When averaged over all samples this score gives a good indication of the performance of the network. The inception score of the classical and quantum-assisted versions (with sparse latent spaces) were ~ 5.7 and ~ 5.6, respectively.

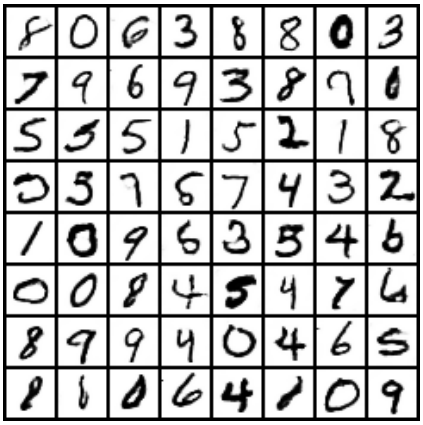
The FID measures the similarity between features extracted by an inception network from the dataset X and the generated data G. The distribution of the features are modelled as a multivariate Gaussian. Lower FID values mean the features extracted from the generated images are closer those for the real images. In Equation (3.18), μ are the means of the activations of an interim layer of the inception network and Σ are the covariance matrices of these activations. The classical and quantum-assisted algorithms (with sparse latent spaces) scored ~ 29 and ~ 23, respectively.

$$\mathrm{FID}(X,G) = ||\mu_X - \mu_G||_2^2 + \mathrm{Tr}\left(\Sigma_X + \Sigma_G - 2\sqrt{\Sigma_X\Sigma_G}\right) \tag{3.18}$$

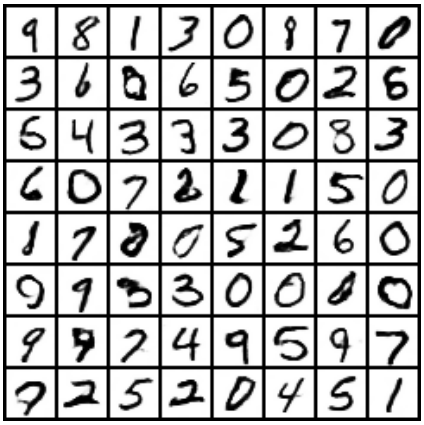
The classical implementation, with a sparse latent space, was also used to generate images mimicking the LSUN bedrooms dataset, Figure 3.9. The Large Scale scene UNderstanding (LSUN) [293] dataset are images of 10 scenes, where there are on average a million examples of each scene. From this dataset we only took examples of one scene: Bedrooms. This image set had around 300,000 images. This final experiment was only performed as a demonstration of scalability, and no metrics were used to evaluate performance.

3.4.1 Discussion

Though it is trivial to demonstrate a correlation between the connectivity of a graphical model and the quality of the learned model, Figure 3.7, it is not immediately clear that the benefits of



((a))



((b))

Figure 3.8: Example MNIST characters generated by (a) classical and (b) quantum-assisted associative adversarial network architectures, with sparse topology latent spaces.

increasing the complexity of the latent space can be detected easily in deep learning frameworks, such as the quantum-assisted Helmholtz machine [29] and those looking to exploit quantum models [129]. The effect of the complexity of the latent space model on the quality of the final latent variable generative model was not apparent in our investigations. Deep learning frameworks looking to exploit quantum hardware supported training in the latent spaces need to truly benefit from this application, and not iron out any potential gains with backpropagation. For example, if exploiting a quantum model gives improved performance on some small test problem, it is an open question as to whether this improvement will be detected when integrated into a deep learning framework, such as the architecture presented here.

Here, given the nature of the demonstration and a desire to avoid chaining we use a sparse connectivity model. Avoiding chaining allows for larger models to be embedded into near-term quantum hardware. Given the $O(n^2)$ scaling of qubits to logical variables for a complete logical graph [62], future applications of sampling via quantum annealing will likely exploit restricted graphical models. Though the size of near-term quantum annealers has followed Moore's law trajectory, doubling in size every two years, it is not clear what size of probabilistic graphical models will find mainstream usage in machine learning applications and exploring the uses of different models will be an important theme of research as these devices grow in size.

There are two takeaways from the results presented here. Though these values are not comparable to state-of-the-art GAN architectures and are on a simple MNIST implementation, they serve the purpose of highlighting that the inclusion of a near-term quantum device is not detrimental to the performance of this algorithm. Secondly, we have demonstrated the framework on the larger, more complex, dataset LSUN bedrooms, Figure 3.9. This indicates that the algorithm can be scaled.

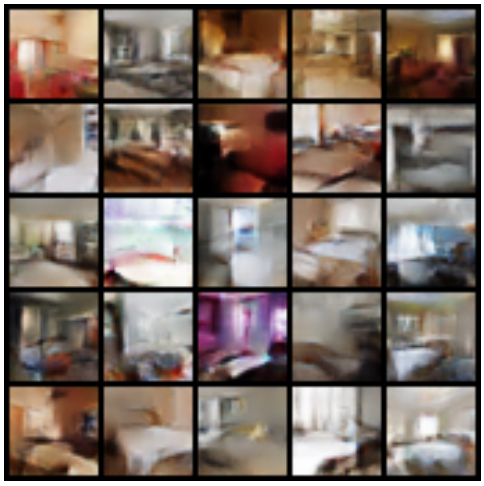


Figure 3.9: Bedrooms from the LSUN dataset generated with an associative adversarial network, with a fully connected latent space sampled via MCMC sampling.

3.5 Conclusions

3.5.1 Summary

In this work we have presented a novel and scalable quantum-assisted algorithm, based on a GAN framework, which can learn a implicit latent variable generative model of complex datasets.

This work is a step in the development of algorithms that may use quantum phenomena to improve the learning generative models of datasets. This algorithm fulfills the requirements of the three areas outlined by Perdomo-Ortiz *et al* [200]: Generative problems, data where quantum correlations may be beneficial, and hybrid. This implementation also allows for use of sparse topologies, removing the need for chaining, requires a relatively small number of variables (allowing for near-term quantum hardware to be applied) and is resistant to noise.

Though the key motivation of this work is to demonstrate a functional deep learning framework integrating near-term quantum hardware in the learning process, it builds on classical work by Tarik Arici and Asli Celikyilmaz [18] exploring the effect of learning the feature space and using this distribution as the input to the generator. No claims are made here on the improvements that can be made classically, though it is possible that further research into the associative adversarial architecture will yield improvements to GAN design.

In summary, we have successfully demonstrated a quantum-assisted GAN capable of learning a model of a complex dataset such as LSUN, and compared performance of different topologies.

3.5.2 Limitations

Throughout the paper several limitations to quantum annealing devices and the methods of integrating quantum annealing devices to neural network models have been mentioned. We

collect these ideas here to highlight the areas in which quantum annealers may have to improve to increase the usefulness of these methods.

One area is being able to understand and benchmark these models against classical counterparts [265], which is already a difficult task in classical machine learning. We believe better benchmarking tools are required to begin evaluating the effectiveness of quantum models. The difficulty of benchmarking is a known problem [219] and there are many criteria to evaluate these devices, for example in the field of optimization the 'time-to-target' metric [130] or number of calls [266]. Here, we do not evaluate our model by other metrics, preferring to restrict the research to establishing a scalable model. We leave the task of evaluating this model by other criteria to future work.

The cost of embedding a logical graph in a quantum annealer may be large in terms of the performance of the model [160]. Higher connectivity of quantum annealing devices will most likely be needed to continue improving quantum-assisted model performance.

There are several assumptions built into the learning procedure of the Boltzmann machine. We assumed samples from the quantum annealer are from a Boltzmann distribution, which is not necessarily true due to the freeze-out hypothesis [11] (and embedding), and also did not determine the temperature of the quantum annealer, under the gray-box assumption [29]. Better models for the behaviour of the device and potentially heurisitics for mitigating their effects would help further improve and understand these methods.

3.5.3 Further Work

There are many avenues to use quantum annealing for sampling in machine learning, topologies and GAN research. Here, we have outlined a framework that works on simple (MNIST) and more complex (LSUN) datasets. We highlight several areas of interest that build on this work.

The first is an investigation into how the inclusion of quantum hardware into models such as this can be detected. There are two potential improvements to the model: Quantum terms improve the model of the data distribution; or graphical models, which are classically intractable to learn for example fully connected, integrated into the latent spaces, may improve the latent variable generative model learned. Before investing extensive time and research into integrating quantum models into latent spaces it will be important to note that these improvements are reflected in the overall model of the dataset. That is, that backpropagation does not erase any latent space performance gains.

There are still outstanding questions as to the distribution the quantum annealer samples. The pause and reverse anneal features on the D-Wave 2000Q gives greater control over the distribution output by the quantum annealer, and can be used to explore the relationship between the quantum nature of that distribution and the quality of the model trained by a quantum Boltzmann machine [162]. It is also not clear what distribution is the 'best' for learning a model of a distribution. It could be that efforts to decrease the operating temperature of a quantum

annealer to boost performance in optimisation problems will lead to decreased performance in ML applications, as the diversity of states in a distribution decreases and probabilities accumulate at a few low energy states. There are interesting open questions as to the optimal effective temperature of a quantum annealer for ML applications. This question fits within a broad are for research in ML asking which distributions are most useful for ML and why. As larger gate model quantum processors become available, it will be interesting to evaluate a variety of quantum algorithms beyond quantum annealing for sampling in this framework.

For this simple implementation, the quantum sampling sparse graph performance is comparable to the complete and restricted topologies. Though in optimised implementations we expect divergent performance, the sparse graph serves the purpose of demonstrating the QAAAN architecture. Additionally, we have highlighted sparse classical graphical models for use in the architecture demonstrated on LSUN bedrooms. Though they have reduced expressive power there are many more applications for current quantum hardware; for example a fully connected graphical model would require in excess of 2048 qubits (the number available on the D-Wave 2000Q) to learn a model of a standard MNIST dataset, not to mention the detrimental effect of the extensive chains. A sparse D-Wave 2000Q native graph (Chimera) conversely would only use 784 qubits. This is a stark example of how sparse models might be used in lieu of models with higher connectivity. Investigations finding the optimal balance between the complexity of a model, resulting overhead required by embedding, and the affect on both on performance are needed to understand how future quantum annealers might be used for applications in ML. More generally, understanding how the architectural, as well as algorithmic, choices in near-term gate model devices affect performance will enhance our understanding of quantum computings impact on machine learning in the decades ahead.

CHAPTER 4

OPTIMIZING QUANTUM HEURISTICS WITH METALEARNING

You keep using that word,
I do not think it means what you think it means.
— Inigo Montoya*, The Princess Bride*

Variational quantum algorithms, a class of quantum heuristics, are promising candidates for the demonstration of useful quantum computation. Finding the best way to amplify the performance of these methods on hardware is an important task. Here, we evaluate the optimization of quantum heuristics with an existing class of techniques called 'meta-learners'. We compare the performance of a meta-learner to evolutionary strategies, L-BFGS-B and Nelder-Mead approaches, for two quantum heuristics (quantum alternating operator ansatz and variational quantum eigensolver), on three problems, in three simulation environments. We show that the meta-learner comes near to the global optima more frequently than all other optimizers we tested in a noisy parameter setting environment. We also find that the meta-learner is generally more resistant to noise, for example seeing a smaller reduction in performance in *Noisy* and *Sampling* environments and performs better on average by a 'gain' metric than its closest comparable competitor L-BFGS-B. Finally, we present evidence that indicates the meta-learner trained on small problems will generalize to larger problems. These results are an important indication that meta-learning and associated machine learning methods will be integral to the useful application of noisy near-term quantum computers.

4.1 Introduction

Machine learning is a powerful tool for tackling challenging computational problems [36, 152, 258]. A recent explosion in the number of machine learning applications is driven by the availability of data, improved computational resources and deep learning innovations [119, 145, 173]. Inter-

estingly, machine learning has also been applied to the problem of improving machine learning models, in a field known as meta-learning [148, 264].

In general, meta-learning is the study of models which 'learn to learn'. A prominent example of a meta-learner model is one that learns how to optimize parameters of a function [14, 60, 150, 212]. Traditionally, this function might be a neural network [14] or a black-box [60]. Meta-learning and other new methods, including Auto-ML [82], are changing the way we train, use and deploy machine learning models [183, 188, 227]. Here, we use a meta-learner to find good parameters for quantum heuristics, and compare that approach to other parameter optimization strategies.

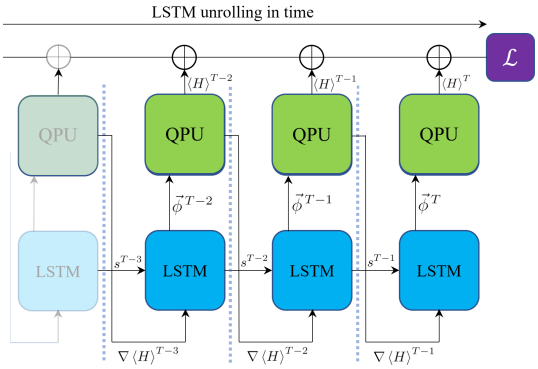


Figure 4.1: Meta-learner training on a Quantum Processing Unit (QPU - green). This diagram illustrates how the meta-learner used in this work can optimize the parameters of a quantum circuit (see Section 4.3 for a full description). Here, we outline a high level description for each time-step, such as $T-2$ (shown). A model, in our case a long short-term memory (LSTM) recurrent neural network (blue) (Section 4.2), takes in the gradients of the cost function. The LSTM outputs parameters $\vec{\phi}$ for the QPU to try at the next step. This procedure takes place over several time-steps in a process known as unrolling. The costs from each time-step are summed to compute the loss, $\mathscr{L}$ (purple), at time T.

Figure 4.1 shows an example of what the implementation of a meta-learner might look like, in the context of optimizing the parameters of a parametrized quantum circuit, illustrated as a quantum processing unit (QPU). In this work, we refer to a QPU and a quantum circuit interchangeably.

Recent progress in quantum computing hardware has encouraged the development of quantum heuristic algorithms that can be simulated on near-term devices [179, 207]. One important heuristic approach involves a class of algorithms known as variational quantum algorithms. Variational quantum algorithms are 'hybrid' quantum-classical algorithms in which a quantum circuit is run multiple times with variable parameters, and a classical outer loop is used to optimize those parameters (see Figure 4.2). The VQE [201], quantum approximate optimization algorithm and its generalization QAOA [79, 104] are examples of algorithms that can be implemented in this variational setting. These algorithms are effective in optimization [98, 190, 215] and simulation of quantum systems [107, 194, 222]. The classical subroutine is an optimization

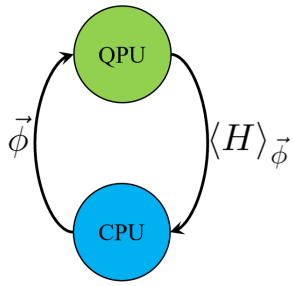


Figure 4.2: A single time-step of a general variational quantum algorithm, where the classical processing unit (CPU - blue) outputs parameters $\vec{\phi}$ dependent on some evaluation, in this case the expectation value $\langle H \rangle$ by the quantum processing unit (QPU - green). The quantum subroutine is encoded by a quantum circuit $U(\vec{\phi})$ (Figure 4.3) parameterized by $\vec{\phi}$, and it is responsible for generating a state $|\psi(\vec{\phi})\rangle$. This state is measured in order to extract relevant information (e.g. expectation value of a Hamiltonian). The classical subroutine suggests parameters $\vec{\phi}$ based on the values provided by a quantum computer, and sends new parameters back to the quantum device. This process is repeated until the given goal is met, i.e. convergence to a problem solution (e.g. the ground state of a Hamiltonian).

of parameters, and is an important part of the algorithm both in terms of the quality of solution found and the speed at which it is found.

Techniques for the classical outer loop optimization are well-studied [98, 99, 185, 201, 272, 273] and several standard optimization schemes can be used. However, optimization in this context is difficult, due to technological restrictions (e.g. hardware noise), and to theoretical limitations such as the stochastic nature of quantum measurements [133] or the *barren plateaus* problem [169]. Therefore, it is imperative to improve not only the quantum part of the hybrid algorithms, but also to provide a better and more robust framework for classical optimization. Here, we focus on the classical optimization subroutine, and suggest meta-learning as a viable tool for parameter setting in quantum circuits. Moreover, we demonstrate that these methods, in general, are resistant to noisy data, concluding that these methods may be especially useful for algorithms implemented with noisy quantum hardware.

We compare the performance of optimizers for parameter setting in quantum heuristics, specifically variational quantum algorithms. The optimization methods we compare are L-BFGS-B [47], Nelder-Mead [187], evolutionary strategies [226] and a Long Short Term Memory (LSTM) recurrent neural network model [110] - the meta-learner. Whilst in the production of this work, we noticed similar research [261] exploring the potential of gradient-free meta-learning techniques as initializers. Here, we use a gradient-based version of the meta-learner as a standalone optimizer (not an initializer), and a larger set of other optimizers. Though we include a diverse range of techniques, clearly, there are other optimizers that might be used, for example SPSA [122, 180, 243, 244], however our analysis focuses on those described above.

This comparison is performed in three different simulation environments: *Wave Function*,

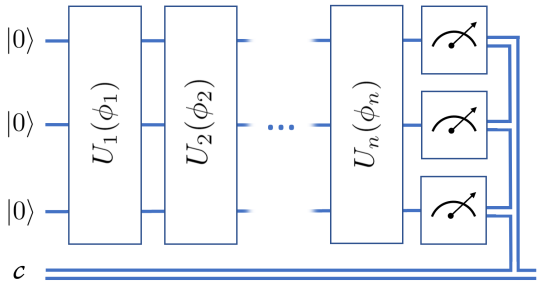


Figure 4.3: General parameterized quantum circuit, with arbitrary unitaries $U_j(\phi_j)$, input state $|\mathbf{0}\rangle$ and classical register $\mathbf{c}$, where $\vec{\phi} = [\phi_1, \phi_2, ..., \phi_n]$ are the parameters of the circuit. Though the unitaries do not necessarily act on all qubits, we have arranged them here in 'blocks', similar to the general architectures of QAOA and VQE, where a block of operations may be repeated many times in a circuit, with different parameters. In the case of VQE, a block might be a series of single qubit rotations or a set of entangling gates (such as CNOT), and for QAOA, a block might be a phase unitary encoding the cost function or a mixing unitary for searching the solution space.

Sampling and *Noisy*. The *Noisy* environment is an exact wave function simulation with parameter setting noise. The simulation environments are defined in detail in Section 4.3.

The first heuristic we explore for this comparison is QAOA [79, 104] for the MAX-2-SAT and Graph Bisection constraint satisfaction problems [196]. Second, VQE [201] is used for estimating the ground-state of Free Fermions models a special subclass of Fermi-Hubbard models [114]. We show that, broadly speaking, the meta-learner performs as well or better than the other optimizers, measured by a 'gain' metric defined in Section 4.4. Most notably, the meta-learner is observed to be more robust to noise. This is highlighted through showing the number of near-optimal solutions found in each problem by the different optimizers over all simulation environments. The takeaway of this paper is that these methods show promise, specifically the features of robustness and adaptability to hardware, and how meta-learning might be applied to noisy near-term devices.

In Section 4.2 we describe the background of the heuristics and optimizers. Then in Section 4.3 we outline the general setup including problems, the optimizers, and the simulation environments. Section 4.4 details the methods, including the metrics, optimizer configuration and meta-learner training. In Section 4.5 we discuss our results. Finally in Section 4.6 the work is summarized and we suggest paths forward.

Contributions

- developed hybrid quantum-classical variational methods integrating classical metalearning with variational quantum algorithms
- compared multiple (L-BFGS-B, Nelder-Mead, evolutionary strategies and metalearning) optimization routines on multiple problems and algorithms
- presented evidence that metalearning may perform better than standard methods especially in environments with parameter setting noise (which will be a feature of near-term hardware)

4.2 Background

4.2.1 Quantum Alternating Operator Ansatz

The quantum approximate optimization algorithm [79] and its generalization the quantum alternating operator ansatz [104] (QAOA) form families of parameterized quantum circuits for generating solutions to combinatorial optimization problems. After initializing a suitable quantum state, a QAOA circuit consists of a fixed number p blocks (see Figure 4.3), where each block is composed of a phase unitary generated from the cost function we seek to optimize, followed by a mixing unitary. The phase unitary typically yields a sequence of multiqubit Pauli-Z rotations each with phase angle γ. In the original proposal of Farhi et al. [79], the mixing unitary is a Pauli-X rotation of angle β on each qubit. However, extending the protocol to more general encodings and problem constraints naturally leads to a variety of more sophisticated families of mixing operators [104, 105]. At the end of the circuit a measurement is performed in the computational (Pauli-Z) basis to return a candidate problem solution.

An important open research area is to develop strategies for determining good sets of algorithm parameters (i.e. the γ and β values for each block) which yield good (approximate or exact) solutions with nonnegligible probability. These parameters may be determined a priori through analysis, or searched for as part of a classical-quantum hybrid algorithm using a variational or other approach. Prior work on parameter setting in QAOA includes analytic solutions for special cases [269], comparison of analytical and finite difference methods [99], a method for learning a model for a good schedule [273], and comparison of standard approaches over problem classes [185].

We evaluate parameter setting strategies for QAOA for MAX-2-SAT and Graph Bisection, both NP-hard combinatorial optimization problems [21, 196]. We use standard‘[79] and generalized [104] QAOA methods, respectively. The latter problem mapping is of particular interest as it utilizes an advanced family of QAOA mixing operators from [104] that has recently been demonstrated to give advantages over the standard mixer [270].

4.2.2 Variational Quantum Eigensolver

The VQE [201] is a hybrid optimization scheme built on the variational principle. It aims to estimate the ground state energy of a problem Hamiltonian through iterative improvements of a trial wave function. The trial wave function is prepared as a quantum state using a parameterized quantum circuit, and the expectation value of the Hamiltonian with respect to this state is measured. This energy value is then passed to a classical device, which uses optimization techniques (SPSA, BFGS, etc.) to update the parameters. The process is repeated for a fixed number of iterations, or until a given accuracy achieved.

The initial demonstration of VQE used Nelder-Mead, a standard derivative-free approach, for parameter setting after observing that gradient descent methods did not converge [201]. Since then, examples in the literature include the use of Simultaneous Perturbation Stochastic Approximation (SPSA) in [180], where the authors argue simultaneous perturbation methods might be particularly useful for fermionic problems, but classical problems (such as MaxCut) may favor more standard techniques (i.e. gradient descent). Other routines used include COBLYA, L-BFGS-B, Nelder-Mead and Powell in [218]. Finally, in [182] the authors explore the use of Bayesian optimization for parameter setting in VQE.

4.2.3 Meta-learning

Meta-learning is the study of how to design machine learning models to learn fast, well and with few training examples [34]. One specific case is a model, referred to here as a meta-learner [212], which learns how to optimize other models. A model is a parameterized function. Meta-learners are not limited to training machine learning models; they can be trained to optimize general functions [60]. In the specific area of using models to optimize other models, early research explored Guided Policy Search [150], which has been superceded by LSTMs [14, 28, 60, 277]. An LSTM is a recurrent neural network, developed to mitigate vanishing or exploding gradients prominent in other recurrent neural network architectures [35, 109]. It consists of a cell state, a hidden state, and gates, and all three together are called an LSTM cell. At each time-step, changes are made to the cell state dependent on the hidden state, the gates (which are models) and the data input to the LSTM cell. The hidden state is changed dependent on the gates and the input. The cell state and hidden state are then passed to the LSTM cell at the next time-step. A full treatment of an LSTM is given in Reference [110]. An LSTM is good for learning long-term (over many time-steps) dependencies, like those in optimization.

Meta-learners have been used for fast general optimization of models with few training examples [212]: Given random initial parameters we seek to achieve a fast convergence to ‘good’ (defined by some metric) general parameters. This same problem feature appears for QAOA, where good parameters may follow some common distribution across problems [273]. A meta-learner could be used to find general good parameters, and fine-tuning left to some other optimizer [260], though this approach was not explored here.

4.3 Setup

4.3.1 Simulation Environments

We compare optimization methods in '*Wave Function*', '*Sampling*' and '*Noisy*' simulation environments. The *Wave Function* case is an exact wave function simulation. For *Sampling*, the simulation emulates sampling from a hardware-implemented quantum circuit, where the variance of the expectation value evaluations is dependent on the number of samples taken from the device. In these experiments, we set the number of shots (samples from the device) to 1024.

Lastly, in the *Noisy* case we have modelled only parameter setting noise in an exact wave function simulation, which is a coherent imperfection resulting in a pure state.[1] We assume exact, up to numerical precision, computation of the expectation value (via some theoretical quantum computer which can compute the expectation value of a Hamiltonian given a state up to arbitrary precision). Then, for each single-qubit rotation gate, we added normally distributed, standard deviation $\sigma = 0.1$, noise to the parameters at each optimization step. In order to determine σ, we evaluated the relationship between the fidelity of an arbitrary rotation (composed of three single-qubit Pauli rotation gates $R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$), around the Bloch sphere and parameter noise; see Figure 4.5. Assuming industry standard single qubit gate rotations of 99% [136], a value of $\sigma = 0.1$ is approximated, see Figure 4.4. All simulations were performed with Rigetti Forest [1] simulators and circuit simulations performed on an Intel(R) Core(TM) i7-8750H CPU with 6 cores.

[1] Other noise models, such as dephasing or depolarizing noise generate mixed states, which is out of scope of this paper. Additionally, it is argued that the optimal set of circuit parameters has similar form for noisy and noiseless cases, for more information see for example [163, 290]

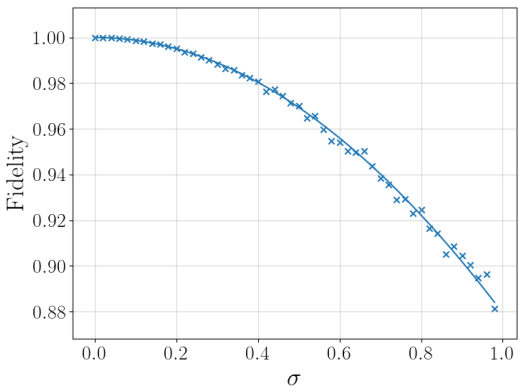


Figure 4.4: Effective single qubit rotation gate fidelity plotted as a function of the noise on input parameters. Parameters are sampled from a normal distribution with standard deviation σ and centered on the target input value.

4.3.2 Optimizers

4.3.2.1 Local optimizers

Nelder-Mead and L-BFGS-B are gradient-free and gradient-based approaches, respectively, which are standard local optimizers [99, 185, 272, 273]. Local optimizers have a notion of location in the solution space. They search for candidate solutions from this location. They are usually fast, and are susceptible to finding local minima. L-BFGS-B is a local optimizer and has access to the gradients. Out of all optimizers chosen it is the closest to the meta-learner in terms of information available to the optimizer and computational burden (i.e. the cost of computing the gradients). Nelder-Mead was chosen as it appears throughout the literature [99, 201, 218, 260] and provides a widely recognized benchmark.

4.3.2.2 Evolutionary Strategies

Evolutionary strategies are a class of global black-box optimization techniques: A population of candidate solutions (individuals) are maintained, which are evaluated based on some cost function. Genetic algorithms and evolutionary strategies have been used for decades. More recent work has shown these techniques to be competitive in problems of reinforcement learning [226, 263].

All implementations of evolutionary strategies are population-based optimizers. In the initial iteration, the process amounts to a random search. In each iteration, solutions with lower costs are more likely to be selected as parents (though all solutions have a nonzero probability of selection). Different methods for selecting parents exist, but we used binary tournament selection, in which two pairs of individuals are selected, and the individual with the lowest cost from each

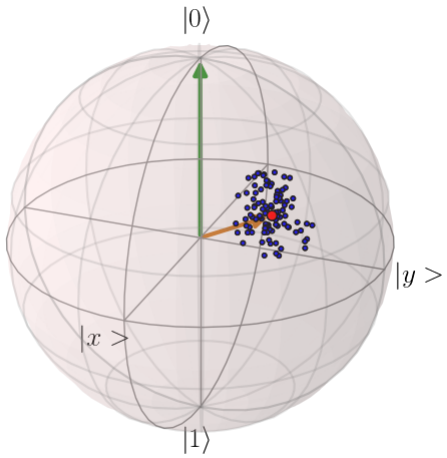


Figure 4.5: Rotation of initial state $|0\rangle$ (green) by rotation operator $R_Z(\pi/4)R_Y(\pi/3)R_Z(0)$ to new state (orange arrow, red point). When noise of $\sigma = 0.1$ is applied to the parameter setting we see a distribution of final states (blue) over 100 trials.

pair is chosen to be a parent.

In more precise terms, parents are the candidate solutions selected to participate in crossover. Crossover takes two parent solutions and produces two children solutions by randomly exchanging the bitstring defining the first parent with the second. Each child replaces its parent in the population of candidate solutions. The process is repeated, so costs for each child are evaluated, and these children are used as parents for the next iteration [27]. In our case, the bitstring is divided into n subsections, where n is the number of parameters passed to the quantum heuristic. Each subsection is converted to an integer using Gray encoding and then interpolated into a real value in the range $[-\pi/2, \pi/2]$. Gray codes are used as they avoid the Hamming walls found in more standard binary encodings [59].

It is the bitstrings that are operated on by the genetic algorithm. When two individuals are selected to reproduce, a random crossover point, b_c is selected with probability P_c. Two children are generated, one with bits left of b_c from the first parent and bits to to the right of b_c originating from the second parent. The other child is given the opposite arrangement. Intuitively, if b_c is in the region of the bitstring allocated to parameter ϕ_k, the first child will have angles identical to the first parent before ϕ_k and angles identical to the second parent after ϕ_k. Again, the second child has the opposite arrangement. The effect on parameter ϕ_k is more difficult to describe. Finally, after crossover is complete, each bit in each child's bitstring (chromosome) is then flipped (mutated) with probability P_m. Mutation is useful for letting the algorithm explore candidate solutions that may not be accessible through crossover alone.

Evolutionary strategies are highly parallelizable, robust and relatively inexpensive [226] making them a good candidate for the optimization of quantum heuristics.

4.3.2.3 Meta-learning on quantum circuits

The meta-learner used in this work is an LSTM, shown unrolled in time in Figure 4.1. Unrolling is the process of iteratively updating the inputs, x, cell state and hidden state, referred to together as s, of the LSTM. Inputs to the model were the gradients of the cost function w.r.t. the parameters, preprocessed by methods outlined in the original work [14]. At each time-step they are

$$x^t = \begin{cases} (\frac{\log(|\nabla \langle H \rangle^t|)}{r}, \text{sign}(\nabla \langle H \rangle^t)) & \text{if } \left|\nabla \langle H \rangle^t\right| \geq e^{-r} \\ (-1, \exp(r) \nabla \langle H \rangle^t), & \text{otherwise} \end{cases} \tag{4.1}$$

where r is a scaling parameter, here set to 10, following standard practice [14, 212]. The terms $\nabla \langle H \rangle^t$ are the gradients of the expectation value of the Hamiltonian at time-step t, with respect to the parameters $\vec{\phi}^t$. This preprocessing handles potentially exponentially-large gradient values whilst maintaining sign information. Explicitly, the meta-learner used here is a local optimizer. At some point $\vec{\phi}^t$ in the parameter-space, where t is the time-step of the optimization, the gradients x^t are computed and passed to the LSTM as input. The LSTM outputs an update $\Delta\vec{\phi}^t$, and the new point in the parameter space is given by $\vec{\phi}^{t+1} = \vec{\phi}^t + \Delta\vec{\phi}^t$. It is possible to use these models for

derivative-free optimization [60], however, given that the gradient evaluations can be efficiently performed on a quantum computer, scaling linearly with the number of gates, and that the optimizers usually perform better with access to gradients, we use architectures here that exploit this information. In Reference [169] the authors show that the gradients of the cost function of parameterized quantum circuits may be exponentially small as a function of the number of qubits, the result of a phenomena called the concentration of quantum observables. In cases where this concentration is an issue, there may be strategies to mitigate this effect [94], though it is not an issue in the small problem sizes used here.

Though only one model (a set of weights and biases) defines the meta-learner, it was applied in a '*coordinatewise*' way: For each parameter a different cell state and hidden state of the LSTM are maintained throughout the optimization. Notably, this means that the size of the meta-learning model is only indirectly dependent on the number of parameters in the problem. We used a gradient-based approach, exploiting the parameter-shift rule [229] for computing the gradients of the loss function with respect to the parameters. These were used at both training and test time.

All model training requires some loss function. We chose the summed losses,

$$\mathscr{L}(\omega) = \mathbb{E}_f \left[\sum_{t=0}^{T} \omega_t f(\phi_t) \right], \tag{4.2}$$

where $\mathbb{E}_f$ is the expectation over all training instances f and T is a time-horizon (the number of steps the LSTM is unrolled before losses from the time-steps $t < T$ are accumulated, backpropagated, and the model parameters updated). The hyperparameters ω_t are included, though are set to $\omega_t = 1$ for all t in these training runs. This can be adjusted to weigh finding optimal solutions later in the optimization more favourably, a practice for balancing exploitation and exploration. In situations where exploration is more important, other loss functions can be used, such as the expected improvement or observed improvement [60]. However, in this instance we chose a loss function to rapidly converge, meaning fewer calls to the QPU. This has the effect of converging to local minima in some cases, though we found that this loss function performed better than the other gradient-based optimizer (L-BFGS-B) for these problems.

4.3.3 Problems

4.3.3.1 Free Fermions Model

Hubbard Hamiltonians have a simple form, as follows:

$$\begin{aligned} H = &- t \sum_{\langle i,j \rangle} \sum_{\sigma=\{\uparrow,\downarrow\}} (a^{\dagger}_{i,\sigma} a_{j,\sigma} + a^{\dagger}_{j,\sigma} a_{i,\sigma}) \\ &+ U \sum_{i} a^{\dagger}_{i,\uparrow} a_{i,\uparrow} a^{\dagger}_{i,\downarrow} a_{i,\downarrow} - \mu \sum_{i} \sum_{\sigma=\{\uparrow,\downarrow\}} a^{\dagger}_{i,\sigma} a_{i,\sigma}, \end{aligned} \tag{4.3}$$

where $a^{\dagger}_{i,\sigma}, a_{i,\sigma}$ are creation and annihilation operators, respectively, of a particle at site i with spin σ. In this model there is a hopping term t, a many body interaction term U and an

onsite chemical potential term μ. This model gained importance as being a possible candidate Hamiltonian to describe superconductivity in cuprate materials. However, recent numerical studies have shown that there are some significant differences between the model and what is seen in experiments, such as the periodicity of charged stripes that the model supports [112, 144, 231]. However, the model is quite interesting itself, with many different phases of interest. The model is also quite difficult to solve, especially when going to large lattice sizes and large values of U/t. This has lead to many studies and much method development on classical computers, and is still widely researched today.

For VQE we look for the ground-state of the simplified spinless three-site Free Fermions model with unequal coupling strengths $t_{ij} \in [-2,2]$ and $U = \mu = 0$, Figure 4.6. The Hamiltonian of this model can be mapped through the Jordan-Wigner transformation [120] to the qubit Hamiltonian

$$\begin{aligned} H_{FH} = \frac{1}{2}\Big(& t_{12}\hat{X}_1\hat{X}_2 + t_{12}\hat{Y}_1\hat{Y}_2 + t_{23}\hat{X}_2\hat{X}_3 \\ & + t_{23}\hat{Y}_2\hat{Y}_3 + t_{13}\hat{X}_1\hat{Z}_2\hat{X}_3 + t_{13}\hat{Y}_1\hat{Z}_2\hat{Y}_3\Big) \end{aligned} \tag{4.4}$$

where $\hat{X}$, $\hat{Y}$ and $\hat{Z}$ are the Pauli-X, Y and Z matrices, respectively. Based on the results of [285, 286], we use a circuit composed of 3 blocks. Each block consists of three single qubit rotations $R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$ applied to all qubits, followed by entangling CNOT gates acting on qubits (1,2) and (2,3), where the first entry is the control qubit and the second is the target.

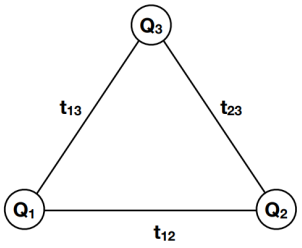


Figure 4.6: Sketch of a spinless three-qubit Free Fermions model that is used for the VQE optimization. Coupling strengths are not necessarily equal and take values from $[-2,2]$.

4.3.3.2 MAX-2-SAT

Given a Boolean formula on n variables in conjunctive normal form (i.e. the AND of a number of disjunctive two-variable OR clauses), MAX-SAT is the NP-hard problem of determining the maximum number of clauses which may be simultaneously satisfied. The best classical efficient algorithm known achieves only a constant factor approximation in the worst case, as deciding whether a solution exists that obtains better than a particular constant factor is NP-complete [196]. For MAX-2-SAT, where each clause consists of two literals, the number of satisfied

clauses can be expressed as

$$C = \sum_{(i,j)\in E} \tilde{x}_i \vee \tilde{x}_j \tag{4.5}$$

where $\tilde{x}_i$ in each clause represents the binary variable x_i or its negation, and E is the set of clauses. We use an n-qubit problem encoding where the jth qubit logical states $|0\rangle_j, |1\rangle_j$ encode the possible values of each x_j. Transforming to Ising spin variables [103] and substituting with Pauli-Z matrices leads to the cost Hamiltonian

$$\widehat{C} = \sum_{(i,j)\in E} \frac{1}{4}(1 \pm \hat{Z}^{(i)})(1 \pm \hat{Z}^{(j)}) \tag{4.6}$$

which is minimized when the number of satisfied clauses is maximized. The sign factors $+1$ or -1 in $\widehat{C}$ correspond to whether each clause contains x_i or its negation, respectively. Note that C and $\widehat{C}$ are *not* equivalent; C gives a maximisation problem, while $\widehat{C}$ gives a minimization problem, with the same set of solutions.

For our QAOA implementation of MAX-2-SAT we use the original [79] initial state $|s\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$, phase operator $U_P(\widehat{C}, \gamma) = \exp\left(-i\gamma\widehat{C}\right)$, and mixing operator $U_M(\beta) = \exp\left(-i\beta \sum_{j=1}^{n} \hat{X}^{(j)}\right)$. The instances we consider below have $n = 8$ qubits, 8 clauses, and QAOA circuit depth $p = 3$. We further explore instances with $n = 12$ and $p = 5$, Figure 4.9.

4.3.3.3 Graph Bisection

Given a graph with an even number of nodes, the Graph Bisection problem is to partition the nodes into two sets of equal size such that the number of edges across the two sets is minimized. The best classical efficient algorithm known for this problem provably yields only a log-factor worst-case approximation ratio [137]. Both this problem and its maximization variant are NP-hard [196].

For an n-node graph with edge set E we encode the possible node partitions with n binary variables, where x_j encodes the placement of the jth vertex. In this encoding, from the problem constraints the set of feasible solutions is encoded by strings x of Hamming weight $n/2$. The cost function to minimize can be expressed as

$$C = \sum_{(i,j)\in E} \mathrm{XOR}(\mathrm{x_i}, \mathrm{x_j}) \tag{4.7}$$

under the condition $\sum_{j=1}^{n} x_j = n/2$. Transforming again to Ising variables gives the cost Hamiltonian

$$\widehat{C} = \frac{1}{2} \sum_{(i,j)\in E} (1 - \hat{Z}^{(i)}\hat{Z}^{(j)}). \tag{4.8}$$

A mapping to QAOA for this problem was given in [104, App. A.3.2] from which we derive our construction. We again encode possible partitions x with the n-qubit computational basis states

$|x\rangle$. For each problem instance we uniformly at random select a string y of Hamming weight $n/2$ and use the feasible initial state $|y\rangle$. The phase operator $U_P(\widehat{C},\gamma) = \exp\left(-i\gamma\widehat{C}\right)$ is constructed in the usual way from the cost Hamiltonian. For the mixing operator we employ a special case of the XY-mixer proposed in [104]. This class of mixers affect state transitions only between states of the same Hamming weight, which will importantly restrict the quantum state evolution to the feasible subspace. For each node $j = 1,\ldots,n$, we define the XY partial mixer

$$U_j(\beta) = \exp\left(-i\beta\left(\hat{X}^{(j)}\hat{X}^{(j+1)} + \hat{Y}^{(j)}\hat{Y}^{(j+1)}\right)\right)$$

with $\sigma^{(n+1)} := \sigma^{(1)}$. We define the overall mixer to be the ordered product $U_M(\beta) = U_n(\beta)\ldots U_2(\beta)U_1(\beta)$. Observe that as each partial mixer preserves feasibility, so does $U_M(\beta)$, and so QAOA will only output feasible solution samples. We consider problem instances with $n = 8$ qubits, 8 edges, and QAOA circuit depth $p = 3$.

4.4 Methods

4.4.1 Metrics

Here, we outline two metrics used to evaluate and compare the optimizers. The first metric used is the gain, $\mathscr{G}$, to the minimum,

$$\mathscr{G} = \mathbb{E}_f\left[\frac{f_F - f_I}{f_{\min} - f_I}\right] \tag{4.9}$$

where $\mathbb{E}_f$ is the expectation value over all instances f, f_F is the converged cost of the optimizer, f_I is the initial cost (determined by the initial parameters) and $f_{\min}$ is the ground-state energy. $f_{\min}$ was determined by evaluating all possible solutions in the cases of MAX-2-SAT and Graph Bisection, and by exact diagonalization of the Hamiltonian for finding the ground-state of the Free Fermions model. This number is the expectation over instances f of the 'gain' to the global minimum from the initialized parameters. In the case of local optimizers (meta-learner, L-BFGS-B, Nelder-Mead) we initialized to the same parameters. The metric outlines the average progress to the global minimum from an initialization. Secondly, the quality of the final solution was also evaluated by a distance to global minima metric, $\mathscr{D}$,

$$\mathscr{D} = \frac{|f_{\min} - f_F|}{|f_{\min} - f_{\max}|} * 100 \tag{4.10}$$

where $f_{\max}$ is the maximum possible energy. This metric gives a sense of the closeness to the global minima, as a percentage of the extent.

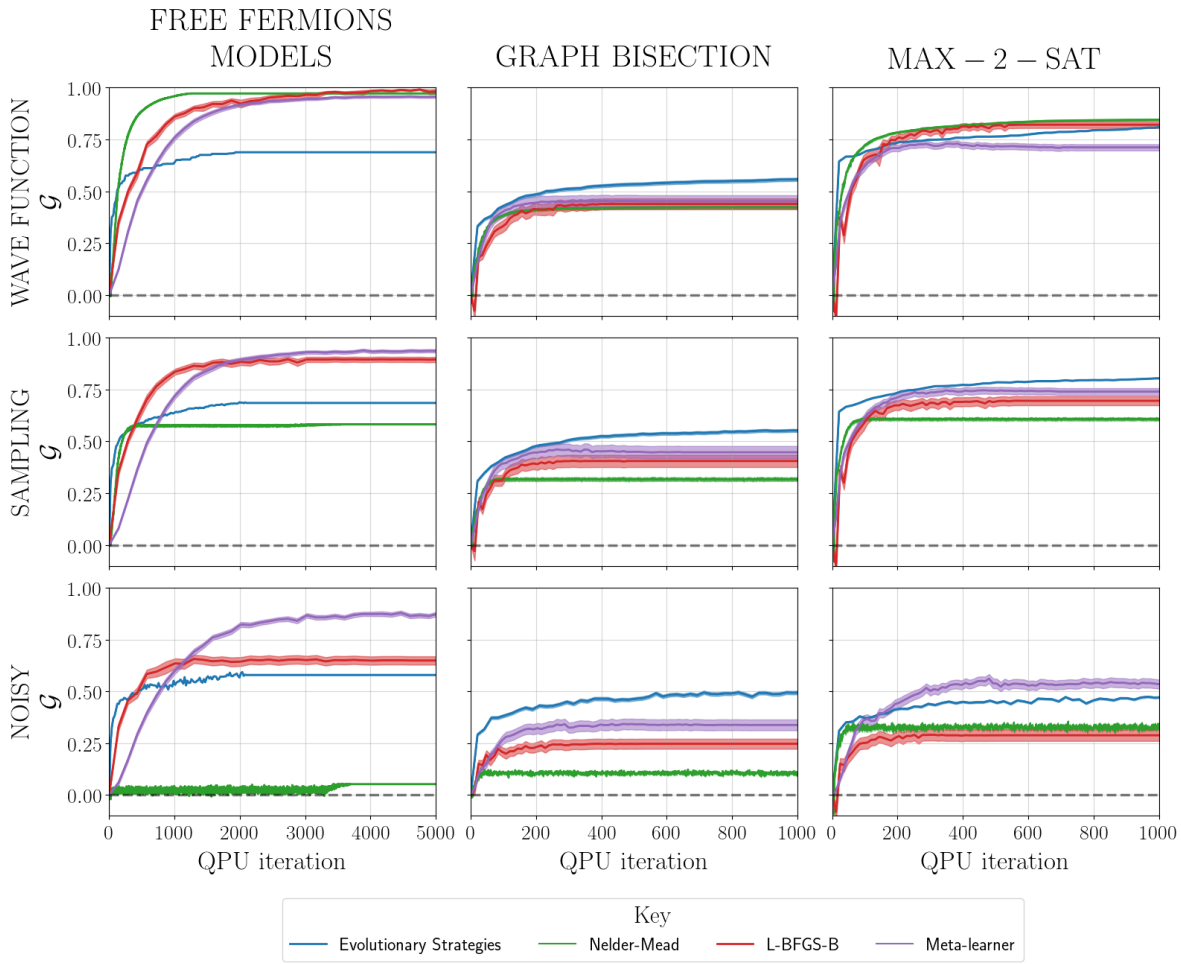


Figure 4.7: Left to right columns: Free Fermions models, Graph Bisection and MAX-2-SAT problems. Top to bottom rows: *Wave Function*, *Sampling* and *Noisy* simulations, defined in Section 4.3. Optimizers: Evolutionary strategies (blue), Nelder-Mead (green), L-BFGS-B (red), meta-learner (purple). x-axis: Shared within a column, QPU iteration is number of calls to the QPU. y-axis: Shared within a row, $\mathcal{G}$, the gain, is the value computed by Equation (4.9), and represents the average progress toward the minimum from the initial evaluation of $\langle H \rangle$. L-BFGS-B and the meta-learner have access to the gradient, and make numerous calls to auxiliary quantum circuits (simulated in the same environment as the expectation value evaluation circuits) to compute the gradients. The number of calls to evaluate gradients of parameters is $N_g = 2M$, where M is the number of parameterized gates in the circuit. The QPU iteration variable captures this, i.e. is the total number of calls to a QPU for an optimizer. Error bars are the standard error on the mean, $\sigma_f/\sqrt{n}$ where n is the number of examples and σ_f the standard deviation of the performance of the optimizers. Note that negative values of $\mathcal{G}$ are observed, corresponding to on average performing worse than the initial evaluation.

4.4.2 Configuring Optimizers

We evaluated the optimizers on 20 problems from 5 random initializations each, to increase the probability of reaching the ground-state by all optimizers. The initializations were kept the same between the local optimizers (L-BFGF-B, Nelder-Mead and meta-learner).

Evolutionary strategies used 5 different random initializations for each problem. L-BFGS-B and Nelder-Mead were implemented using Scipy [118], where the gradients for L-BFGS-B were computed by analytic means and quantum circuit simulation. We implemented and configured the evolutionary strategies methods in-house. For all tests, a small population size of 20 was used to limit the number of calls to the simulator (sizes on the order of 100 are typical and may improve performance). Both MAX-2-SAT and Graph Bisection problems with QAOA used $m = 60$ bits to represent parameters. VQE simulations had more parameters to optimize, so $m = 297$ bits were used for these problems. All tests used a probability of crossover of $P_c = 0.9$, and a probability of mutation of $P_m = 0.01$. These parameters were selected by a sparse grid search.

On these small problems, the SciPy default hyperparameters of standard optimizers L-BFGS-B and Nelder-Mead were found to give generally good performance. Any tuning did not contribute meaningfully to the performance, though we expect at larger problem sizes more tuning will be required as the optimization landscape increases in ruggedness. We found these hyperparameters generalized well.

4.4.3 Training the meta-learner

For the MAX-2-SAT and Graph Bisection problems the model was trained on just 200 problems, whereas in the case of optimizing Free Fermions models the meta-learning model quickly converged and training was truncated at 100 problems. The loss function is given in Equation (4.2), where values $\omega_t = 1 \; \forall \; t$ are used. For the preprocessing of the gradients, the hyperparameter r in Equation (4.1) is set to 10. For all training an Adam optimizer [131] was used with a learning rate of 0.003, $\beta_0 = 0.9$, $\beta_1 = 0.999$, $\epsilon = 1.0^{-8}$ and zero weight decay. These training schedules were consistent across simulation type (*Wave Function*, *Sampling* and *Noisy*). We included a 'curriculum' method, implemented in [60], whereby the time-horizon of the meta-learner is extended slowly throughout the training cycle. This was started at 3 iterations and capped at 10, at the end of the training cycle. Optimization was terminated if it converged, under standard convergence criteria. Overall, 9 models were trained (3 simulation environments × 3 problem classes).

4.5 Discussion & Results

Figure 4.7 shows the performance of the optimizers measured by the gain metric in the three simulation environments. The gain metric converges in the same sense as an optimizer converging on one problem instance, this is as expected given it is an average over many problem instances. A value close to 1 is desirable, indicating the ability of an optimizer to progress to the global

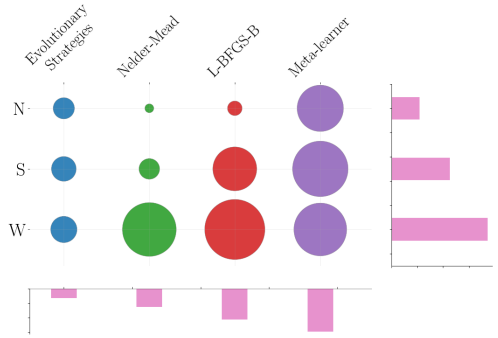


Figure 4.8: Bubble and bar plots of the frequency of near-optimal solutions. The size of each bubble is dependent on the total number of times an optimizer came within 2% of the global optima across all problem instances (computed by Equation (4.10)); the largest bubble is L-BFGS-B in the *Wave Function* environment (115). Repetitions are included, i.e. if an optimization ended in a near-optimal solution it was counted, regardless of whether it was found in a previous optimization. We found that if one optimizer performed well in one task, it performed well, relative to the other optimizers, in another (by this metric), so each bubble is not divided into each problem class. The right bar plot represents the summation across optimizers within a simulation type. The bottom bar plot represents the summation within an optimizer across simulation types. (N - *Noisy*, S - *Sampling*, W - *Wave Function*)

minima from a starting point. Figure 4.8 shows the total number of near-optimal solutions found by each optimizer. We define near-optimal as finding a solution within 2% of the global optima computed by Equation (4.10). The closest comparable competitor to the meta-learner in these plots is L-BFGS-B, given both optimizers had access to the gradients. This is reflected in their performance, particularly in Figure 4.7.

It is important to recognize that the comparison in Figure 4.7 has limited scope. Optimization is a hard problem: There are many ways to improve application specific performance of different algorithms and metrics to evaluate that performance. For example, the gradient-based optimizers (meta-learner and L-BFGS-B) evaluate auxiliary quantum circuits many times in order to compute the gradients. Recognizing there are always limitations to comparing optimization methods, we draw conservative conclusions. Additionally, BFGS was not designed to work in noisy evaluation environments and as such is not expected to compete with specially designed optimisers (like the meta-learner).

4.5.1 General Performance

Additionally to meta-learning functioning as an optimizer in variational quantum algorithms, we find competitive performance of this meta-learning algorithm, at small instance size, over a range of problem classes, using the gain metric $\mathscr{G}$ defined in Equation (4.9); see Figure 4.7.

The metric $\mathscr{G}$ was used to evaluate and compare the optimizers, though this value can

hide significant features. For example, an optimizer that finds good (but not optimal) solutions frequently will perform better than an optimizer that finds bad solutions frequently and optimal solutions infrequently. There are other cases that the reader may have in mind. This particular example is addressed in Figure 4.8. The number of times the optimizer comes within 2% of the ground-state (across all problems), as calculated by Equation (4.10), is counted. We observe an expected reduction in performance as noise is increased; this is discussed further in the subsection below.

4.5.2 Noise

As expected, there is a reduction in performance for all optimizers as 'noise' increases: Performance is worse in *Sampling* than in *Wave Function* and is worse in *Noisy* than in *Sampling*. What is notable is that the meta-learner is more resilient to this increase in noise than other methods. For example, in Free Fermions model problems, L-BFGS-B performance reduces by 0.35 whereas the meta-learner only reduces by 0.2, from around the same starting point (Free Fermions models column, Figure 4.7). This pattern is repeated across problem classes, to varying degrees. We believe this is a promising sign that meta-learning will be especially useful in noisy near-term quantum heuristics implemented on hardware. In the case of simulation, we believe this resistance can be explained by the optimizer knowing how to find generally good parameters, having learned from noisy systems already. This needs to be distinguished from another potential benefit of these algorithms, where the models learn how to optimize in the presence of hardware-specific traits. In the latter case, the meta-learner may learn a model that accounts for hardware specific noise. Further, in Figure 4.8 we see a reduction in performance, measured by the total number of near-optimal solutions, for all optimizers. However, this effect is least apparent in the global optimizer (evolutionary strategies) and the meta-learner. Additionally, the meta-learner finds significantly more near-optimal solutions (80) for *Noisy* simulation than the next best optimizer (evolutionary strategies - 17). These are promising results on the potential use cases of these optimizers in hybrid algorithms implemented on noisy quantum hardware.

4.5.3 Evolutionary Strategies

Evolutionary strategies exhibit an oscillatory behavior when gain to global optima versus function call is plotted, the first generation corresponds to a random search, then the fittest individual (i.e. best solution) found in the previous generation is evaluated first in the next generation. Hence, we observe a spike in performance every 21 evaluations (the size of the population plus the fittest individual). As such we only plot every 21 iterations, giving a smooth curve. We reiterate here that given other performance/time metrics, including for example if optimizers are parallelized, other analysis including different comparison metrics will be needed to determine the respective use cases of meta-learners vs evolutionary strategies. Indeed, while Figure 4.7 suggests that evolutionary strategies perform well for particularly hard problems (Graph Bisection, *Noisy*),

preliminary results in Figure 4.8 indicate that the meta-learner tends to outperform evolutionary strategies when searching for a near-optimal solution.

4.5.4 Problems and algorithms

The Free Fermions models were the simplest to solve (they are small problems confined to parameter values [-2,2]). This is reflected in the performance of the gradient-based optimizers. Evolutionary strategies underperforms. This is most likely a result of the size of the parameter space: Though the problem size (in terms of the number of variables) is smaller, there are significantly more parameters in this implementations we have considered of VQE (24) than QAOA (6).

Of the two classical optimization problems we consider, the Graph Bisection problem is harder than MAX-2-SAT, in the sense of worse classical approximability. While MAX-2-SAT can be approximated up to a constant factor, the best classical efficient algorithms known for Graph Bisection perform worse with increasing problem size [21, 196]. This contrast appears in the performance of all optimizers: In general, every optimizer performs worse in Graph Bisection than in MAX-2-SAT by the gain metric.

4.5.5 Scaling

Figure 4.9 provides evidence that the meta-learner model may be generalized. A model trained on smaller QAOA problem instances ($n = 8$, $p = 3$) is extended to larger problems ($n = 12$, $p = 5$). We chose L-BFGS-B for this comparison as it is the closest comparable competitor in terms of information available and performance. The meta-learner is competitive with or even better than L-BFGS-B, as evaluated by the Gain metric, in the initial optimization though appears to have worse asymptotic behaviour. This may be because the the meta-learner encourages large steps in the initial optimization, where the margin for error on the step is larger than when further in the minima. At a high-level the initial and final steps can be thought of as regions with distinct properties, it is unsurprising the meta-learner performs differently in each region.

This small demonstration is not extensive enough to make any serious conclusions regarding the generalization of the meta-learner for optimizing quantum circuits, though it indicates similar findings in the field that these models can extend to larger system sizes [14].

4.6 Conclusion

In this work we compared the performance of a range of optimizers (L-BFGS-B, Nelder-Mead, evolutionary strategies and a meta-learner) across problem classes (MAX-2-SAT, Graph Bisection and Free Fermions Models) of quantum heuristics (QAOA and VQE) in three simulation environments (*Wave Function*, *Sampling* and *Noisy*). We highlight three observations. The first is that the meta-learner outperforms L-BFGS-B (the closest comparable competitor) in most cases, when

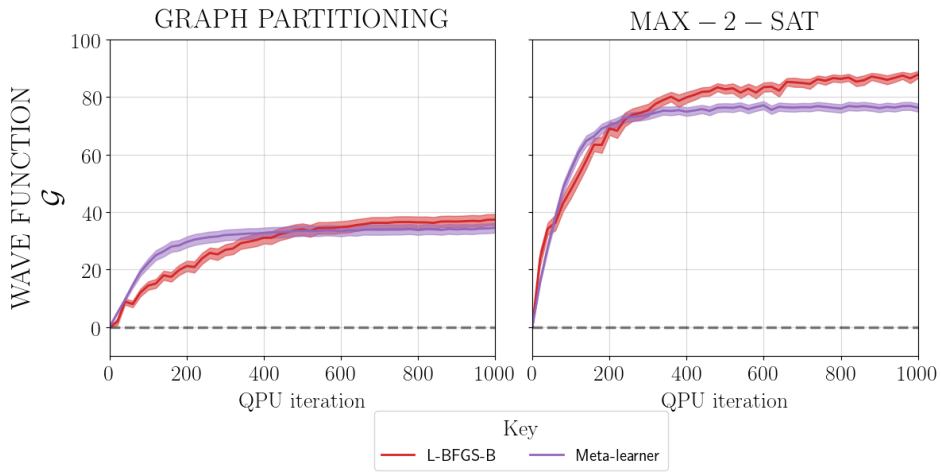


Figure 4.9: Gain to minimum of L-BFGS-B and meta-learner optimizers in a *Wave Function* environment applied to QAOA problems Graph Bisection and MAX-2-SAT. These problems are 12 variable problems with QAOA hyperparameter p=5. This is contrasted with the problems explored in Figure 4.7, which are 8 qubit problems with p=3. The meta-learner is the model trained on this previous problem set. QPU iteration is the number of calls made to a quantum circuit. In this case, each optimization step is $N_g = 2M = 20$, where $M = 10$.

measured by an average percent gain metric $\mathscr{G}$. Secondly, the meta-learner performs better than all optimizers in the *Noisy* environment, measured by a total number of near-optimal solutions metric $\mathscr{D}$. Finally, the meta-learner generalizes to slightly larger systems for QAOA problems, which reflects other findings in the field. We conclude that these are promising results for the future applications of these tools to optimizing quantum heuristics, because these tools need to be robust to noise and we are often looking for near-optimal solutions.

During the production of this work a related preprint [261] was posted online. In that preprint, the authors consider only gradient-free implementations of meta-learners. Their training set is orders of magnitude larger, as the meta-learner is learning to optimize from more limited information. However, taking into account the QPU calls required to compute the gradients, $N_g = 2M$ where M is the number of parameterized gates, their gradient-free implementation required significantly fewer queries to a QPU during optimization. As both architectures have different advantages and trade-offs between resource overhead, training time and performance should be considered for a given use case. Their conclusions are similar to ours regarding the potential of meta-learning methods, and suggest using them as an initialization strategy.

The meta-learning methods evaluated here are relatively new and are expected to continue to improve in design and performance [277]. There are several paths forward, we highlight some here. Though there is no investigation into the scaling of meta-learner performance to larger problem sizes, this in part is limited by the inability to simulate large quantum systems quickly, and exacerbated by the further burden of computing the gradients. It is an open question as

to how meta-learners will perform with quantum heuristics applied to larger problem sizes. In a closely related vein, these methods will be explored on hardware implementations, for two reasons. The first is that quantum computing will soon be beyond the realm of reasonable simulation times, and testing these algorithms on systems with higher number of variables will have to be done on hardware. The second is that these meta-learners may be able to learn hardware-specific features. For example, in this work the meta-learner is a single model applied to different parameters. This approach is called 'coordinatewise'. If instead applied in a 'qubitwise' fashion, where different models are trained for parameters corresponding to each qubit in a given hardware graph, there may be local variability in the physics of each qubit that the meta-learner accounts for in its model and optimization.

In terms of further investigations into the specifics of the problems and quantum heuristics considered, we emphasize that our QAOA implementation of Graph Bisection used a different type of mixer and initial state than MAX-2-SAT. An important question to answer is to what degree the differences in performance we observed between MAX-2-SAT and Graph Bisection are due to the change of mixer and initial state, as opposed to the change of problem structure. Additional possible mixer variants and initial states for Graph Bisection are suggested in [104], which we expect to further affect QAOA performance, and hence also affect the performance of our parameter optimization approaches. An important open area of research is to better characterize the relative power of different QAOA mixers and the inherent tradeoffs in terms of performance, resource requirements, and the difficulty of finding good algorithm parameters. In this direction, recent work [270] has demonstrated that superposition states may perform better than computational basis states as QAOA initial states.

Finally, heuristics play a prominent role in solving real-world problems: They provide practical solutions - not necessarily optimal - for complex problems (where an optimal solution is prohibitively expensive), with reasonable amount of resources (time, memory etc.). Therefore, we see significant potential for applications of quantum heuristics, implemented not only on near-term quantum devices - especially for variational quantum algorithms - but also for hybrid computing in fault-tolerant architectures. Thus it is imperative to characterize the classical components, such as the meta-learner, that learn properties of quantum devices towards the deployment of effective quantum heuristics for important practical applications.

CHAPTER 5

NEURAL NETWORK ANSÄTZE

Your assumptions are your windows on the world.
Scrub them off every once in a while, or the light won't come in.
— Isaac Asimov

This Chapter is dedicated to a short, hopefully illuminating, trip through some methods in quantum chemistry, specifically quantum Monte Carlo, developed to model fermionic wave functions. Broadly, everything discussed is practically useful, and concepts that may be historically significant may be ignored and detours into the theoretical undergrowth are generally avoided. This trip ends in the region directly relevant to work in this Thesis: The application of neural networks to the problem of modelling fermionic wave functions. I will argue, beyond recent state-of-the-art results, outlined in Chapter 6, neural networks will likely become a common if not *de facto* tool for modelling wave functions. The discussion is restricted to systems of fermions (i.e. electrons) to build the core concepts required to describe the work in Chapter 6, though these methods generalize to systems of bosons.

This section draws from many sources. Excellent summaries, discussion and reviews can be found in the literature on the topics of quantum chemistry [85, 89, 106, 249].

5.1 Notation and Prerequisites

5.1.1 The system

The systems explored here are electrons moving in real continuous 3-D Euclidean space aorund nuclei, alternately called fermionic systems in the wave mechanics formulation of quantum mechanics. Electrons are fermions and a fermion is a spin-$\frac{1}{2}$ particle. Systems of fermions are described by an antisymmetric wave function (the sign changes under the exchange of particles) resulting in the Pauli exclusion principle: They cannot occupy the same configuration simulta-

neously. The quantum state of the electrons and nuclei is represented by a wave function $\psi(X)$, where X is the configuration of the system $\{r_1, r_2, ..., r_n, R_1, R_2, ..., R_m\}$. The time-independent Schrödinger equation

$$\hat{H}\psi(X) = E\psi(X). \tag{5.1}$$

governs the spectrum of possible eigenstates. All instances of the Schrödinger equation used in this Chapter are expressed in atomic units (see Section 5.1.2 for a full derivation). The operator on the left-hand side of the equation is called the Hamiltonian, and for the time-independent non-relativistic case is given by

$$\hat{H} = \hat{K}_{\mathrm{e}} + \hat{K}_{\mathrm{n}} + V(X) \tag{5.2}$$

where $\hat{K}_{\mathrm{e}}$ is the kinetic energy operator of the electrons, $\hat{K}_{\mathrm{n}}$ is the kinetic energy operator of the nuclei and $V(X)$ is a potential energy term resulting from the Coulomb interaction of charged particles

$$V(X) = \sum_{i<j}^{n_{\mathrm{e}},n_{\mathrm{e}}} \frac{1}{r_{ij}} + \sum_{i,I}^{n_{\mathrm{e}},n_{\mathrm{n}}} \frac{Z_I}{r_{iI}} + \sum_{IJ} \frac{Z_I Z_J}{R_{IJ}} \tag{5.3}$$

n_{e} is the number of electrons, n_{n} the number of nuclei, Z_I the charge of nucleus I, $r_{ij} = \left|\mathbf{r}_i - \mathbf{r}_j\right|$, $r_{iI} = |\mathbf{r}_i - \mathbf{R}_I|$ and $R_{IJ} = |\mathbf{R}_I - \mathbf{R}_J|$, with $|\cdot|$ denoting the L2-norm of a vector. In general, we take $\mathbf{r}$ and $\mathbf{R}$ to be the position vectors of electrons and nuclei, respectively.

This can be simplified via the Born-Oppenheimer approximation: nuclei in the system are static. Given that the mass of an electron is significantly less than the mass of a nucleus $m_e/m_n = 10^{-3} - 10^{-5}$, the dynamics of the nuclei are ignored and they are fixed in place.

This assumption has two effects. First, the kinetic energy contributions of the nuclei are zero. Second, the contributions to the potential energy from the nucleus-nucleus Coulomb interactions are constant. They can then be neglected; their only effect is to shift the eigenspectrum of the Hamiltonian, which is trivial to recover.

The kinetic energy operator of the electrons $\hat{K}_{\mathrm{e}} = -\frac{\hat{\nabla}^2}{2}$ and $\hat{\nabla}^2$ is the Laplacian operator $\hat{\nabla}^2 = \sum_{i=1}^{n} \sum_{j=x,y,z} \frac{\partial^2}{\partial^2 r_{i,j}}$. Including the potential terms from electron-electron and electron-nucleus Coulomb interactions we can rewrite the Schrödinger equation as

$$\hat{H} = \hat{K}_{\mathrm{e}} + V(X) = -\frac{\hat{\nabla}^2}{2} + \sum_{i<j}^{n_{\mathrm{e}},n_{\mathrm{e}}} \frac{1}{r_{ij}} + \sum_{i,I}^{n_{\mathrm{e}},n_{\mathrm{n}}} \frac{Z_I}{r_{iI}} \tag{5.4}$$

As we will see, much effort has been expended in understanding how to model wave functions, specifically of interest in this Thesis, those that represent the ground-states, *efficiently* and

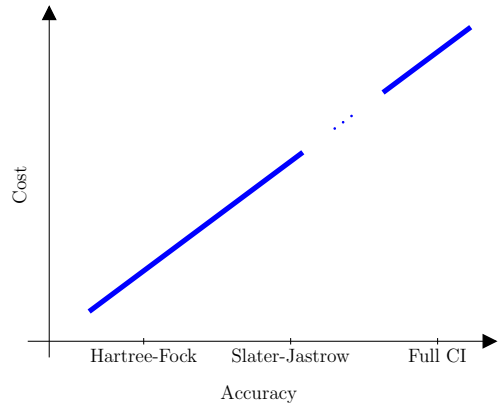


Figure 5.1: Cartoon of a Pople diagram, showing the relationship between the cost and accuracy of the method.

accurately. Any function with the correct properties, for example antisymmetry, is a viable Ansatz; it may, although it will likely not, be a good representation of the ground-state wave function. The model must be able to capture the behaviour of the entire wave function resulting from correlations between particles in the system, where one particle's behaviour is correlated with another. These correlations arise from various sources. In this Chapter, we discuss Fermi correlations, Coulomb correlations and backflow and how models of wave functions are adapted to be able to capture these features *efficiently* and *accurately*, Figure 5.1.

A concrete example of wave function behaviour is a cusp. A cusp in the wave function forms when either an electron approaches the nucleus or another electron and the potential energy term of the Hamiltonian diverges to an infinite value. When the distance is zero the phenomena is referred to as coalescence. This divergence in the potenial is offset by oppositely diverging kinetic energy, creating a cusp in the wave function. A cusp condition is the behaviour the wave function must obey at these points, such as the Kato cusp [125].

As expected, the cost of finding a good Ansatz increases with the desired accuracy. Figure 5.1 is a simplified Pople diagram, outlining the relationship between the computational cost of a method and its ability to capture the behaviour of wave functions. As we will see, a Hartee-Product does not capture any of the electron-electron correlations and is a good starting point for most other methods explored here. In general, chemical accuracy, 1 kcal/mol or 4 kJ/mol, is enough to address most applications of wave function modelling, but an order of magnitude more is needed to study other phenomena such as superconductivity.

5.1.2 Atomic units

All the derivations and equations in this Chapter are written in atomic units. Here, the transformation from SI (International System of Units) units to atomic units is outlined. In SI units the

time-independent non-relativistic Schödinger equation is written

$$\Big[\frac{-\hbar^2}{2m_e}\hat{\nabla}^2 - \frac{e^2}{4\pi\epsilon_0|\mathbf{r}_i - \mathbf{r}_j|}\Big]\psi = E\psi \tag{5.5}$$

by letting all $\mathbf{r}_n \to \lambda\mathbf{r}'_n$ this becomes

$$\Big[\frac{-\hbar^2}{2m_e\lambda^2}\hat{\nabla}'^2 - \frac{e^2}{4\pi\epsilon_0\lambda|\mathbf{r}'_i - \mathbf{r}'_j|}\Big]\psi'(X) = E\psi'(X) \tag{5.6}$$

the constants can be factored when

$$\frac{\hbar^2}{m_e\lambda^2} = \frac{e^2}{4\pi\epsilon_0\lambda} = E_a \tag{5.7}$$

where E_a is a unit of energy called a *Hartree*. Solving for λ

$$\lambda = \frac{4\pi\epsilon_0\hbar^2}{m_e e^2} = a_0. \tag{5.8}$$

λ is the Bohr radius a_0. Finally,

$$E_a\Big[\frac{-1}{2}\hat{\nabla}'^2 - \frac{1}{|\mathbf{r}'_i - \mathbf{r}'_j|}\Big]\psi'(X) = E\psi'(X) \tag{5.9}$$

$$\Big[\frac{-1}{2}\hat{\nabla}'^2 - \frac{1}{|\mathbf{r}'_i - \mathbf{r}'_j|}\Big]\psi'(X) = (E/E_a)\psi'(X) \tag{5.10}$$

Energies computed via the this dimensionless Schrödinger equation are given in Hartrees.

5.2 Wave function models

5.2.1 The wave function

The time-independent wave function in position representation is a complicated function over 3-D Euclidean space mapping configurations to amplitudes. Having access to a model wave function called an Ansatz, which is a good approximation to the ground-state, is useful. These Ansatz are inspired by the physics of the systems and constrained by available computational power. Some Ansatz provide high accuracy but require a prohibitive amount of compute. Some work in certain situations but not in others. As such, determining an easy to find, flexible and accurate Ansatz is, without hyperbole, of paramount importance.

The structure of the Ansatz is important. For reasons that will become obvious, the Ansätze discussed here have structures in common, such as the orbital. An orbital in position representation, denoted as a coordinate dependent function $\phi(\mathbf{r})$, is a wave function for a single electron.

Atomic orbitals are single electronic wave functions in systems with one nucleus, and molecular orbitals are the obvious generalization to systems with more than one nucleus. Depending on context, 'spatial' and 'molecular' are implied throughout.

In all areas discussed in the following sections the wave functions are assumed to be real. For a real Hermitian operator (e.g. the Hamiltonian of atomic systems in real space) the ground-state wave function can be modelled with a real-valued function. There are cases where the wave function will need to be modelled with a complex function, for example Hermitian operators with complex valued eigenstates, in these cases the Ansatz will need to be complex in order to accurately model the wave function. This design choice is limiting in the number of different systems that can be modelled and the quantitites that may be computed from the Ansatz. However, complex numbers are not a mature feature in most state-of-the-art deep learning frameworks. In order to solve problems in quantum chemistry to the extent of incumbent methods, complex deep learning practise first need to become more established.

5.2.2 Hartree product

A Hartree product is a separable state of single particle orbitals,

$$\psi_{\mathrm{H}}(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n) = \phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2)...\phi_n(\mathbf{r}_n) \tag{5.11}$$

where $\psi_{\mathrm{H}}(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n)$ is the Hartree product wave function, and $\phi_1(\mathbf{r}_1)$ is the single particle orbital of particle i. The single particle orbitals are orthonormal

$$\int \phi_i(\mathbf{r})^* \phi_j(\mathbf{r}) d\mathbf{r} = \delta_{ij} \tag{5.12}$$

but practically finite, incomplete in continuous space for systems with more than 1 particle, and the probability density associated with the orbital of an particle is independent of the other particle positions

$$\left|\psi_{\mathrm{H}}(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n)\right|^2 d\mathbf{r} = \left|\phi_1(\mathbf{r}_1)\right|^2 d\mathbf{r}_1 \left|\phi_2(\mathbf{r}_2)\right|^2 d\mathbf{r}_2 ... \left|\phi_n(\mathbf{r}_n)\right|^2 d\mathbf{r}_n. \tag{5.13}$$

Modelling the state as separable results in a mean-field approximation to the Coulomb potential. The particles are assumed to be moving in a self-consistent field. The field is referred to as self-consistent because the single particle wave functions are an eigenfunction of the Hamiltonian and also generate the potential field in which they move.

The overall wave function is *not* antisymmetric, and therefore is not appropriate for modelling systems of fermions. The Hartree-Fock method solves this problem by arranging these orbitals into a Slater determinant.

5.2.3 Slater Determinant

A determinant is a scalar quantity associated with matrix. It is by construction an antisymmetric function; if the rows or columns are exhanged the sign on the determinant is flipped. Concretely,

$$\det(A) = |A| = \begin{vmatrix} A_{00} & \dots & A_{0n} \\ \vdots & & \vdots \\ A_{n0} & \dots & A_{nn} \end{vmatrix} = \sum_{i=0}^{n!} (-1)^{p_i} \mathscr{P}_i A_{00} \dots A_{nn} \tag{5.14}$$

where $\mathscr{P}_i$ is a permutation operator which permutes the column indices and the sum runs over all combinations of the indices. p_i is the number of permutations required to restore a given permutation $i_0, i_1, \dots, i_n$ to the original order $0, 1, \dots, n$

For clarity, an explicit 2x2 example is

$$|A| = \begin{vmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{vmatrix} = (-1)^0 A_{00} A_{11} + (-1)^1 A_{01} A_{10} \tag{5.15}$$

A Slater determinant is a determinant of electron orbitals. These orbitals are usually but not necessarily orthogonal.

5.2.4 Hartree-Fock

Hartree-Fock orbitals are an optimal, in the sense of quality of overall approximation to the ground-state wave function, set of uncorrelated single electron orbitals *in a given basis*. The basis determines the span of the Hilbert space available to the orbitals. The Hartree-Fock wave function is these orbitals arranged in a Slater determinant, enforcing the antisymmetry of the fermionic wave function and introducing Fermi correlations (exchange effects). Given some basis of orbitals $\{\chi_i(\mathbf{r}) | i = 1, 2, \dots, k\}$ spanning a subspace of Hilbert space (but the entire continuum), we can construct an arbitrary set of eigenfunctions

$$\phi_i(\mathbf{r}) = \sum_{j=1}^{k} \alpha_{ij} \chi_{ij}(\mathbf{r}) \tag{5.16}$$

of the Fock operators, which are single electron energy operators that only estimate the electron-electron interations as a mean-field whose sum is an approximation to the true Hamiltonian,

$$\hat{F}_i \phi_i(\mathbf{r}_i) = e \phi_i(\mathbf{r}_i) \tag{5.17}$$

where e is the eigenvalue of $\hat{F}$ and $\phi_i(r_i)$ are the eigenfunctions. By solving a linearly independent set of eigenvalue equations (Roothaan equations [220]) it is possible to retrieve

all α_{ij} which determine the limited basis set of the eigenfunctions of the Fock operator. The eigenvalues, energies, of these orbitals can be ordered. The lowest n_e orbitals are filled with electrons and the unfilled orbitals are referred to as virtual orbitals. These filled orbitals are arranged in a Slater determinant

$$\psi(X) = \det(\mathbf{\Phi}(X)) = \begin{vmatrix} \phi_1(\mathbf{r}_1) & \dots & \phi_n(\mathbf{r}_1) \\ \vdots & & \vdots \\ \phi_1(\mathbf{r}_n) & \dots & \phi_n(\mathbf{r}_n) \end{vmatrix}, \tag{5.18}$$

where the columns correspond to orbitals and the rows to electrons positions. The minimum size of the basis set is equal to the number of electrons in the system. As the basis set increases, the quality of the approximation improves. Configuration interaction methods use a linear combination of Slater determinants to approximate the correlations in the system

$$\psi(X) = \sum_k \omega_k \det(\mathbf{\Phi}_k(X)), \tag{5.19}$$

where $\mathbf{\Phi}_k(X)$ are referred to as configuration state functions. This method is called Full Configuration Interaction (Full CI), if, given a basis, all possible configuration state functions are contained in the wave function. The number of possible Slater determinants is factorial in the size of the basis. As a result, this method is intractable for all but the smallest systems, though it is exact within the basis set chosen. Two n-electron Slater determinants having different orthonormal orbitals (from the same basis set) are also orthogonal.

5.2.5 Slater-Jastrow

The Jastrow factor is a symmetric function introduced into general many-body problems to model particle-particle correlations [116]. The first instance of this for atomic and molecular systems was put forward by Boys and Handy [41, 228].

The general form of the Jastrow factor with 1 and 2 electron terms is

$$J(X) = \sum_i^{n_\mathrm{e}} g(\mathbf{r}_i, \mathbf{R}) - \frac{1}{2} \sum_{i>j,\sigma\mu} u_{\sigma\mu}(\mathbf{r}_i, \mathbf{r}_j) \tag{5.20}$$

where σ and μ are the spin of electron i and j, respectively, $g(\mathbf{r}_i, \mathbf{R})$ describes the electron-nuclear correlations and is a function of the nuclei positions $\mathbf{R}$, and $u(\mathbf{r}_i, \mathbf{r}_j)$ describe the electron-electron correlations.

Given the diverse range of systems it is not in general known *a priori* which Jastrow factor is best in a particular circumstance. There are constraints on the structure of the functions $J_{\sigma\mu}$. In order to maintain the anti-symmetry of the wave function they must be *even* (symmetric)

under the exchange of electrons. Also, they must be twice-differentiable everywhere, other than at $r_{ij} = 0$. In periodic systems, the Jastrow factor must satisfy the boundary conditions of the cell. Finally, Jastrow factor must obey the electron-electron cusp conditions

$$\left.\frac{\partial u_{\sigma\mu}(\mathbf{r}_i, \mathbf{r}_j)}{\partial r_{ij}}\right|_{r_{ij}=0} = \begin{cases} -1/4 & \text{if} \quad \sigma = \mu \\ -1/2 & \text{if} \quad \sigma \neq \mu \end{cases} \tag{5.21}$$

where $r_{ij} = \left|\mathbf{r}_i - \mathbf{r}_j\right|$, giving spherically symmetric behaviour at short distances. A Slater-Jastrow Ansatz is written

$$\psi(X) = e^{J(X)} \sum_k \omega_k \det(\mathbf{\Phi}_k(X)) \tag{5.22}$$

5.2.6 Slater-Jastrow-backflow

The classical notion of backflow is the the movement of a liquid from a system to a source (flowing backwards). For example, water flows down a pipe, introducing an obstruction to the pipe decreases the volume (of the pipe) increases the pressure and results in backflow (flow towards the source).

The quantum analogue is a the phenomena of negative flow of probability (amplitude) in the wave function. The effect was outlined by Feynman and Cohen [83], who derive a wave function describing liquid Helium. The backflow arises as a factor to accommodate the conservation of momentum (and therefore current) after introducing an excitation (phonon) into the model. For a steady state wave function, there must be zero total current, and therefore the introduction of an excitation must be accompanied by the negative flow of particles.

This effect has been included into Ansatz for some time, and the literature seems to capture all momenta dependent correlations, in the case of the electron-nuceli system, between electrons. In specific terms of this model, the movement of electrons are influenced by the movement of other electrons [111].

Practically, introducing backflow correlations to Ansatz involves replacing particle coordinates with quasiparticles

$$\mathbf{r}'_i = \mathbf{r}_i + \sum_{i \neq j} \eta(r_{ij})(\mathbf{r}_i - \mathbf{r}_j) \tag{5.23}$$

$\eta(r_{ij})$ can be a parameterized function obtained via optimization, for example VMC described in Section 5.3.1. In uniform systems, this only results in replacing the coordinates with these quasiparticle coordinates. The nature of the sum ensures that the pseudo-coordinates are dependent on electron i and all other electrons j.

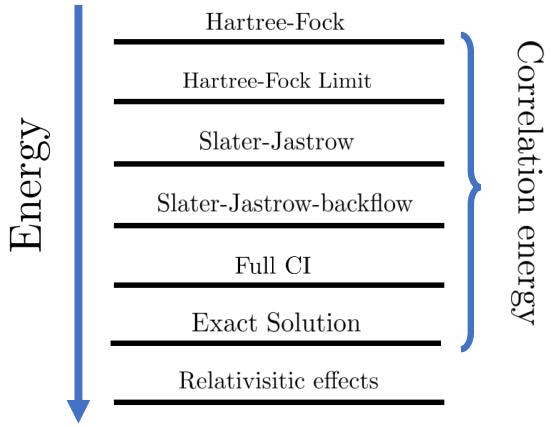


Figure 5.2: Cartoon of the accuracy of the methods described in previous sections. The Hartree-Fock limit is the energy obtained by a complete basis. Relativistic effect shows the solution to the Schrödinger equation when including relativisitic physics into the model and the exact solution is the energy of the exact ground-state of the time independent Schrödinger equation.

5.2.7 Summary

Several methods for modelling wave functions were outlined in the preceding sections. There are scores of methods, approximations and tricks for capturing the behaviour of wave functions. Here, in order to lay the foundation for neural network Ansatz of the form outlined in Chapter 6, we have not touched on important topics such as density functional theory [17] and coupled cluster [195]. They are outside the scope of this work and mentioned here for the interested reader.

Figure 5.2 outlines roughly how the methods outlined in the previous sections compare on an arbitrary scale of approximation to the ground-state energy.

5.3 Quantum Monte Carlo Methods

Quantum monte carlo methods are a broad class of tools for simulating quantum systems effectively defined by the approximation of integrals into terms computable with Monte Carlo methods. For example, the cost of computing the exact partition function scales exponentially with the system size, but can be calculated with statistical uncertainty via sampling.

Take some function $y = f(x)$ of a random variable, where the random variable is distributed according to $p(x)$. A single value drawn from $p(x)$ is called a sample. We can approximate the mean of $f(x)$ by sampling the distribution $p(x)$.

$$\bar{y} = \int p(x)f(x)dx = \mathbb{E}_{x\sim p(x)}[f(x)] \approx \frac{1}{m}\sum_i f(x_i), \tag{5.24}$$

where $\bar{y}$ is the mean of the function. This quantity has some uncertainty associated with it, which is dependent on the number of samples taken from $p(x)$ (and therefore the number of values of $f(x)$ computed),

$$\text{Var}(y) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y})^2 \tag{5.25}$$

In order to compute these quantities, we need to be able to generate samples from $p(x)$ that are representative of the distribution. There are methods designed to perform direct sampling, for example in the case of univariate distributions the inverse probability transform can transform the problem to a uniform sampling problem, which is trivial. Direct sampling methods are often either impossible or impractical if the space is high dimensional.

Markov Chain Monte Carlo (MCMC) methods generate samples by moving walker, creating a chain of states $x_0, x_1, ..., x_m$, around a space in such a way that they eventually represent the target distribution $p(x)$. These samples are correlated, because the new state depends on the previous state and so forth.

Metropolis Hastings is one such algorithm. A walker is a value of the random variable, x, that can change. A typical Monte Carlo simulation will have 100s-1000s of walkers, which we will call a batch in the language of deep learning, but will restrict the following discussion to a single walker for simplicity. A move or a step is a change in the variable to a new value x'. If we have access to a function $g(x) \propto p(x)$ we can compute the acceptance probability of making a move $P_{\text{move}} = \frac{g(x')}{g(x)}$. If the move is accepted, the walker changes to the new value $x \to x'$ and the process is repeated for some number of steps. The Algorithm is detailed in Algorithm 2.

Algorithm 2 Metropolis-Hastings algorithm used here for sampling. c_l is the correlation length. Line 7 updates variables when the condition is true.

1: **for** c_l **do**
2: $\quad \boldsymbol{\xi} \sim N(0, \sigma)$
3: $\quad x' \leftarrow x + \boldsymbol{\xi}$
4: $\quad \alpha \sim U[0, 1]$
5: $\quad P_{\text{move}} \leftarrow \frac{p(x')}{p(x)}$
6: $\quad$ **if** $P_{\text{move}} > \alpha$ **then**
7: $\quad\quad X \leftarrow x'$
8: $\quad$ **end if**
9: **end for**

The autocorrelation length, τ_A, is the number of steps needed to be taken before the samples are independent. Increasing the step size σ and the number of steps c_l decreases the amount correlation between the samples, but in general it is not known what these values should be to completely decorrelate the samples. Therefore given some data we must be able to compute the autocorrelation length to accurately compute quantities such as the mean and the variance. A method for computing the autocorrelation length from a set of samples is outlined below.

Usually, the variance decreases with $1/m$, Equation (5.25). The variance is actually computed from the number, m', of *uncorrelated* samples, expressed

$$\mathrm{Var}_{p(x)}[f(x)]/m' = \tau_A \mathrm{Var}_{p(x)}[f(x)]/m \tag{5.26}$$

and for a stochastic process evolving in time this computed as the sum over all steps in the process τ_A is computed

$$\tau_A = \sum_{x=-\infty}^{\infty} \rho(x) \tag{5.27}$$

where $\rho(x)$ is the normalized autocorrelation function of the process. In this case the Markov chain this can be estimated as

$$\tau_A \approx \sum_{i=1}^{m} \sum_{n=1}^{m-i} \frac{1}{n-i}(y_n - \bar{y})(y_{n-t} - \bar{y}). \tag{5.28}$$

5.3.1 Variational Monte Carlo

VMC is a method for finding the ground state of a quantum system. Given a model of the system, the Hamiltonian, $\hat{H}$, the energy of some wave function $\psi(X)$ can be computed

$$E = \frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi\rangle} = \frac{\int \psi^*(X)\hat{H}\psi(X)dX}{\int |\psi(X)|^2 dX} = \frac{\int |\psi(X)|^2 \frac{\hat{H}\psi(X)}{\psi(X)} dX}{\int |\psi(X)|^2 dX}. \tag{5.29}$$

The integral is taken over all configurations, X, in the space. In the numerator on the right hand side for a given configuration we can split the term into two concepts, the probability of observing a configuration $p(X) = \frac{|\psi(X)|^2}{\int |\psi(X)|^2 dX}$ and the local energy, defined in Section 5.3.2.1.

Computing the integral exactly is hard, so we use approximate Monte Carlo methods to sample configurations from the wave function and approximate the true energy

$$E = \mathbb{E}_{X\sim|\psi(X)|^2}\left[E_L(X_i)\right] \approx \frac{1}{m}\sum_{i=1}^{m} E_L(X_i) \tag{5.30}$$

where $E_L(X_i)$ is the local energy of configuration X_i sampled with probability $p(X_i) = \frac{|\psi(X_i)|^2}{\int |\psi(X)|^2 dX}$. An important observation to note about the expectation value in Equation (5.29) is the local energy is constant across all configurations if $\psi(X)$ is an exact eigenfunction of the $\hat{H}$, such that during the variational optimization of the Ansatz the variance of the local energies of walkers will decrease.

Momentarily we will switch to Dirac notation for the derivation of the variation principle, which states that any wave function will have a higher energy than the ground-state wave

function. Given a system with a discrete set of orthonormal, $\langle\psi_i|\psi_j\rangle = \delta_{ij}$, states with corresponding energies $\mathscr{E} = \{E_0, E_1, ..., E_n\}$, which can be ordered $E_0 \leq E_1 \leq, ..., \leq E_n$, and an arbitrary normalized state $\langle\tilde{\psi}\rangle = 1$, we can derive the *variational principle*,

$$\langle\tilde{\psi}|\hat{H}|\tilde{\psi}\rangle \geq E_0$$

First we note

$$\begin{aligned}
\langle\tilde{\psi}\rangle &\overset{\text{NORM}}{=} 1 \\
&\overset{\text{IDEN}}{=} \sum_{ij} \langle\tilde{\psi}|\psi_i\rangle \langle\psi_i|\psi_j\rangle \langle\psi_j|\tilde{\psi}\rangle \\
&\overset{\text{ORTH}}{=} \sum_{ij} \langle\tilde{\psi}|\psi_i\rangle \delta_{ij} \langle\psi_j|\tilde{\psi}\rangle \\
&= \sum_{i} \langle\tilde{\psi}|\psi_i\rangle \langle\psi_i|\tilde{\psi}\rangle \\
&= \sum_{i} \left|\langle\tilde{\psi}|\psi_i\rangle\right|^2, \qquad (5.31)
\end{aligned}$$

where NORM, IDEN and ORTH are the normalization, identity and orthogonal conditions, respectively, and

$$\left\langle\psi_i\middle|\hat{H}\middle|\psi_j\right\rangle = E_j \langle\psi_i|\psi_j\rangle = \delta_{ij} E_j \qquad (5.32)$$

then we show

$$\begin{aligned}
\langle\tilde{\psi}|\hat{H}|\tilde{\psi}\rangle &= \sum_{ij} \langle\tilde{\psi}|\psi_i\rangle \langle\psi_i|\hat{H}|\psi_j\rangle \langle\psi_j|\tilde{\psi}\rangle \\
&= \sum_{i} E_i \left|\langle\tilde{\psi}|\psi_i\rangle\right|^2 \\
&\geq E_0 \qquad (5.33)
\end{aligned}$$

5.3.2 Solving the Schrödinger equation with Variational Monte Carlo

The Schrödinger Equation plays a central role in the description of quantum behavior of chemical systems. Apart from a handful of analytically solvable models, for example the hydrogen atom [42], one mostly needs to incorporate approximate techniques and numerical methods that scale unfavorably with the increasing system size (e.g. number of electrons) [255]. Many techniques and approximations have been introduced to address this problem, and in this work we have a particular focus on real space Monte Carlo approaches.

An atomic or molecular system can be described by the time-independent Schrödinger equation

$$\hat{H}\psi(X) = E\psi(X). \tag{5.34}$$

Using the Born-Oppenheimer approximation, the position of nuclei are frozen and we have a Hamiltonian for the electronic degrees of freedom,

$$\hat{H} = -\frac{1}{2}\hat{\nabla}^2 + V(X). \tag{5.35}$$

Units of energy E expressed here are Hartrees, $\hat{\nabla}^2$ is the multidimensional ($3n_e$ where n_e is the number of electrons) Laplacian of the wave function $\hat{\nabla}^2 = \sum_{i=1}^{n_e}\sum_{j=x,y,z}\frac{d^2}{dr_{i,j}^2}$ that describes kinetic energy. $r_{i,j}$ and $R_{i,j}$ correspond to coordinate $j = x,y,z$ of the i-th electron and nuclei, respectively. $V(X)$ is the potential energy of some (electron, nuclei) configuration

$$X = (r_{1,x}, r_{1,y}, r_{1,z}, \ldots, r_{n_e,z}; R_{1,x}, \ldots, R_{n_n,z}), \tag{5.36}$$

given by

$$\begin{aligned} V(X) = &\sum_{i>j}^{n_e}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{i,I}^{n_e,n_n}\frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} \\ &+ \sum_{I>J}^{n_n}\frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} \end{aligned} \tag{5.37}$$

where n_n is the number of nuclei, Z_I is the atomic number of nuclei I, and $\mathbf{r}_i$ and $\mathbf{R}_I$ are the position vectors of the electron i and nuclei I, respectively. Throughout the Chapter we use bold font to denote vectors and regular font to denote vector components (scalars). Both are lower case symbols or letters. Matrices are capitalized symbols or letters and are not written in bold font. Though a batched vector, for example m vectors of dimension n arranged in an $m \times n$, can be represented as a matrix, we notationally treat it as a vector, as is typical for deep learning. There are some caveats to these, though in general the meaning is obvious from context, for example nuclei coordinates are capitalized.

Solving the Schrödinger equation, Equation (5.34), is a subroutine in finding the minimum energy of the system, i.e. the ground state energy E_0. Since the Hamiltonian is a bounded operator, one may use a variational principle [249]

$$E_0 \leq \frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi\rangle} = \frac{\int dX \psi^*(X)\hat{H}\psi(X)}{\int dX \psi^*(X)\psi(X)}, \tag{5.38}$$

detailing that the expectation value of the Hamiltonian $\hat{H}$ with respect to a state $\psi(X)$ is bounded from below by the ground state energy E_0. Finding the best approximation to the ground state $\psi_0(X)$ can be done with a parameterized Ansatz, a so-called trial wave function $\psi(X;\theta)$, which is iteratively optimized until a satisfactory accuracy (in terms of energy) is achieved. Different

Ansätze have varying capacities to express wave functions, resulting in different possible minimal energy wave functions. The greater the capacity of an Ansatz to model the true wave function, the better the approximation to the ground state, in general.

A popular class of variational methods - Variational Monte Carlo (VMC) - relies on random sampling of the configuration space in order to estimate expectation value of the Hamiltonian (computing loss function) as

$$\mathscr{L}(\theta) = \frac{\langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle}{\langle\psi(\theta)\rangle} = \frac{\int dX|\psi(X;\theta)|^2 E_L(X;\theta)}{\int dX|\psi(X;\theta)|^2}, \tag{5.39}$$

where $E_L(X;\theta) = \psi^{-1}(X;\theta)\hat{H}\psi(X;\theta)$ is local energy, which for molecular/atomic Hamiltonians is convenient to express in log-domain as

$$E_L(X';\theta) = -\frac{1}{2}\left[\hat{\nabla}^2 \log|\psi(X;\theta)|\big|_{X'} + \left(\hat{\nabla}\log|\psi(X;\theta)|\big|_{X'}\right)^2\right] + V(X'). \tag{5.40}$$

$(\hat{\nabla}\cdot)^2$ is the inner product of the nabla operator $\left(\frac{\partial\cdot}{\partial r_{1,x}}, ..., \frac{\partial\cdot}{\partial r_{n_e,z}}\right)$ with itself. The integral (5.39) is an expectation value of the sampled configurations X,

$$\int dX|\psi(X;\theta)|^2 E_L(X;\theta) = \mathbb{E}_{X\sim p(X;\theta)}\Big[E_L(X;\theta)\Big]. \tag{5.41}$$

The expectation in Equation (5.41) is approximated by a Monte-Carlo estimate,

$$\mathbb{E}_{X\sim p(X;\theta)}\Big[E_L(X;\theta)\Big] \approx \frac{1}{N}\sum_{i=1}^{N} E_L(X_i;\theta), \tag{5.42}$$

where we introduce configuration probability $p(X;\theta) \propto |\psi(X;\theta)|^2$. Samples (also referred to as walkers and are represented by X) are generated from the wave function distribution via the Metropolis Hastings Monte Carlo method. In order to update the parameters θ and improve the wave function, one needs to compute gradients of the loss function with respect to θ denoted $\Delta\mathscr{L}(\theta)$. The parameters θ of the wave function are optimized using some form of gradient descent and computed via

$$\Delta\mathscr{L}(\theta) = \mathbb{E}_X\left[(E_L(X;\theta) - \mathbb{E}_X[E_L(X;\theta)])\hat{\nabla}\log|\psi(X;\theta)|\right] \tag{5.43}$$

and estimated through sampling of the configuration space. This procedure allows us to get close to the ground state $\psi_0(X)$, however it strongly relies on the parameterized Ansatz and ease of computing the gradients $\Delta\mathscr{L}(\theta)$. From now on, we will omit θ parameters where it is clear from the context, and introduce Fermi Net as an Ansatz that provides powerful parameterization.

5.3.2.1 Optimization of the parameterized Ansatz

Up to this point we have assumed an arbitrary general state $|\psi\rangle$. Now, we introduce an Ansatz, or model, $\psi(X;\theta)$ parameterized by θ following the guidelines laid out in the previous sections. In

order to improve the Ansatz we must change the parameters in such a way that minimizes the energy. The derivatives of the energy with respect to the parameters are

$$\frac{\partial E}{\partial \theta} = \frac{\partial}{\partial \theta} \frac{\int \left|\psi(X)\right|^2 \frac{\hat{H}\psi(X)}{\psi(X)} dX}{\int \left|\psi(X)\right|^2 dX}, \tag{5.44}$$

which can be simplified by substituting $\frac{\partial \psi(X)}{\partial \theta} = \psi(X) \frac{\partial \log \psi(X)}{\partial \theta}$

$$\begin{aligned} \frac{\partial E}{\partial \theta} &= \frac{\int \left|\psi(X)\right|^2 \frac{\partial \log \psi}{\partial \theta} E_L(X) dX}{\int \left|\psi(X)\right|^2 dX} + \frac{\int \left|\psi(X)\right|^2 E_L(X) dX \int \left|\psi(X)\right|^2 \frac{\partial \log \psi(X)}{\partial \theta} dX}{\int \left|\psi(X)\right|^2 dX} \\ &= \mathbb{E}_{X\sim|\psi(X)^2|}\left[E_L(X) - \mathbb{E}_{X\sim|\psi(X)^2|}[E_L(X)] \frac{\partial \log \psi}{\partial \theta}\right], \end{aligned} \tag{5.45}$$

with the local energy given by

$$E_L(X) = \frac{\hat{H}\psi(X)}{\psi(X)} = -\frac{1}{2}\left[\hat{\nabla}^2 \log|\psi(X)|\big|_X + \left(\hat{\nabla} \log|\psi(X)|\big|_X\right)^2\right] + V(X). \tag{5.46}$$

$(\hat{\nabla}\cdot)^2$ is the inner product of the nabla operator $\left(\frac{\partial \cdot}{\partial r_{1,x}}, ..., \frac{\partial \cdot}{\partial r_{n_e,z}}\right)$ with itself, $\hat{\nabla}^2$ is the Laplacian defined previously and $V(X)$ is the potential energy of the system with fixed nuclei

$$V(X) = \sum_{i<j}^{n_\mathrm{e},n_\mathrm{e}} \frac{1}{r_{ij}} + \sum_{i,I}^{n_\mathrm{e},n_\mathrm{n}} \frac{Z_I}{r_{iI}} \tag{5.47}$$

5.3.3 Diffusion Monte Carlo

Whereas VMC relies on the optimization of the parameters of the Ansatz via the derivatives of the expectation value of the energy, DMC is a projector method relying repeated application of a projector operator. Practically, this involves weighting the significance of the walkers. Given a trial function, which has some non-zero overlap with the ground-sate, repeated application of a projector (via the power method) will evolve the wave function toward the ground-state. Green's function Monte Carlo is an example of a projector method. Diffusion Monte Carlo is a related but distinct projector method that is equivalent to a Green's function method in the limit of small time steps [85, 158, 257].

Attempting to simulate the imaginary-time Schrödinger equation in this way naïvely one would encounter the sign problem [101, 255], resulting from the antisymmetry constraint of exchange of electrons. The most successful approach to avoiding the sign problem in DMC simulations is the fixed node approximation: The nodes of the wave function are fixed [13]. Under this assumption, the nodes (points in space where the wave function changes from positive to

negative) are fixed in place. Practically, this is implemented by not allowing walkers to cross nodes during the sampling process.

Green's function Monte Carlo and DMC are essentially equivalent [147]. Though a complete description of Green's function Monte Carlo is outside the scope of this Thesis, I briefly mention the exact propagator method as a small amount of context to Stochastic Reconfiguration and DMC.

Given some parameterized Ansatz $|\psi(\theta)\rangle$ that is equal to the trial state, we wish to change the wave function such that it is equal to the projected state. We start with the imaginary-time Schrödinger equation

$$-\frac{\partial\psi(X,t)}{\partial t} = (\hat{H} - E_T)\psi(X,t) \tag{5.48}$$

that can be interpreted as a diffusion equation and solved in the path integral formalism [135]. E_T is the trial energy or offset energy, discussed further below. In this equation, the ground-state of the system is a stationary state; there are no time dynamics and the LHS is equal to zero. In this case, we can recover the time-independent Schrödinger equation $\hat{H}\psi(X) = E_0\psi(X)$.

For some state $\psi_n(X) \neq \psi_0(X)$ with eigenvalue $E_n > E_T$ we can see that the derivative will also be negative: The amplitudes of states which have higher eigenvalues than E_T decay as a function of time. It is clearer to see when the wave function is expanded as a linear combination eigenfunctions

$$-\frac{\partial\psi(X,t)}{\partial t} = (\hat{H} - E_T)\sum_i \alpha_i\psi_i(X,t) \tag{5.49}$$

and the eigenfunctions of this equation are

$$\begin{aligned}\psi(X,t) &= e^{-t(\hat{H}-E_T)}\sum_{i=0}^{\infty}\alpha_i\psi_i(X)\\ &= \sum_{i=0}^{\infty}\alpha_i e^{-t(E_i-E_T)}\psi_i(X)\end{aligned} \tag{5.50}$$

in the limit of $t\to\infty$ the behviour of the wave function is dependent on E_T. There are three cases for the behaviour in the asymptotic region:

$$\lim_{t\to\infty}\psi(X,t) = \begin{cases}\infty & \text{for } E_T > E_0\\ \alpha_0\psi_0 & \text{for } E_T = E_0\\ 0 & \text{for } E_T < E_0\end{cases}. \tag{5.51}$$

The trial energy is not known beforehand. It is set adaptively dependent on the behaviour the evolution. In DMC E_T is chosen to make the ground-state energy zero. In Green's function Monte Carlo, it is chosen to make the spectrum of the Hamiltonian positive.

5.3.4 Stochastic reconfiguration

Stochastic reconfiguration was initially suggested as an approach to mitigate the effect of sign problem in lattice Green's function Monte Carlo [241]. It was subsequently developed into an optimization routine for VMC methods [55, 242].

We can absorb the trial energy into the Hamiltonian and express the time-component of the wave function as a Taylor series

$$\begin{aligned} e^{-\tau(\hat{H}-E_T)} &= e^{-\tau\hat{H}'} \\ &= \sum_{k=0}^{\infty} \frac{(-\tau)^k \hat{H}'^k}{k!} \\ &= \hat{I} + \sum_{k=1}^{\infty} \frac{(-\tau)^k \hat{H}'^k}{k!} \\ &= \hat{I} - \tau\hat{H}' + \mathcal{O}(\tau^2) \\ &\approx \hat{I} - \tau\hat{H}'. \end{aligned} \tag{5.52}$$

This is a first order approximation (as the second order terms are dropped) dependent on τ. The task then becomes identifying a way to apply the operator $\hat{I} - \tau\hat{H}'$, which is described below. The states are represented in this derivation with Dirac notation. Application of the projector in Equation (5.52) to some trial Ansatz $|\psi_T\rangle$ returns the projected state $|\psi_P\rangle$

$$|\psi_P\rangle = (\hat{I} - \tau\hat{H})|\psi_T\rangle\,. \tag{5.53}$$

Taking the derivatives of the trial Ansatz with respect to its parameters forms a basis that defines the subspace around the state, where the Ansatz can vary in the directions the parameters control. The parameter derivative operators are $\hat{O}_i|\psi\rangle = \partial|\psi\rangle/\partial\alpha_i = |\psi_i\rangle$ and $\hat{O}_0|\psi\rangle = \hat{I}|\psi\rangle = |\psi\rangle$ [55, 198, 242]. The new Ansatz can be defined in terms of this basis and amplitudes $\delta\alpha_i$, steps in the directions of the derivatives,

$$|\psi'_T\rangle = \delta\alpha_0|\psi_T\rangle + \sum_i \delta\alpha_i\hat{O}_i|\psi_T\rangle \tag{5.54}$$

$$= \sum_i \delta\alpha_i\hat{O}_i|\psi_T\rangle \tag{5.55}$$

Equating the overlap of the states $|\psi_i\rangle = \hat{O}|\psi\rangle$ of the imaginary time evolved state and the updated to the variational Ansatz we have

$$\langle\psi_T|\hat{O}_j^\dagger(\hat{I} - \tau\hat{H})|\psi_T\rangle = \langle\psi_T|\hat{O}_j^\dagger(\sum_i \delta\alpha_i\hat{O}_i)|\psi_T\rangle \tag{5.56}$$

which can be separated into the cases where $j = 0$ and otherwise,

$$1 - \tau \langle \psi_T | \hat{H} | \psi_T \rangle = \delta\alpha_0 + \sum_i \delta\alpha_i \langle \psi_T | \hat{O}_i | \psi_T \rangle \qquad \text{if } j = 0 \tag{5.57}$$

and

$$\left\langle \psi_T | \hat{O}_j^\dagger | \psi_T \right\rangle - \tau \left\langle \psi_T | \hat{O}_j^\dagger \hat{H} | \psi_T \right\rangle = \delta\alpha_0 \left\langle \psi_T | \hat{O}_j^\dagger | \psi_T \right\rangle + \sum_i \delta\alpha_i \left\langle \psi_T | \hat{O}_j^\dagger \hat{O}_i | \psi_T \right\rangle \quad \text{otherwise} \tag{5.58}$$

Substituting the expression for $\delta\alpha_0$ in Equation (5.57) into Equation (5.58) we have

$$\left\langle \hat{O}_j^\dagger \right\rangle - \tau \left\langle \hat{O}_j^\dagger \hat{H} \right\rangle = (1 - \tau \langle \hat{H} \rangle - \sum_i \delta\alpha_i \langle \hat{O}_i \rangle) \left\langle \hat{O}_j^\dagger \right\rangle + \sum_i \delta\alpha_i \left\langle \hat{O}_j^\dagger \hat{O}_i \right\rangle \tag{5.59}$$

$$\tau \langle \hat{H} \rangle \left\langle \hat{O}_j^\dagger \right\rangle - \tau \left\langle \hat{O}_j^\dagger \hat{H} \right\rangle = -\sum_i \delta\alpha_i (\langle \hat{O}_i \rangle \left\langle \hat{O}_j^\dagger \right\rangle + \left\langle \hat{O}_j^\dagger \hat{O}_i \right\rangle) \tag{5.60}$$

where $\left\langle \hat{O}_j^\dagger \right\rangle = \left\langle \psi_T | \hat{O}_j^\dagger | \psi_T \right\rangle$ for compactness of notation. solving this linear system of equations to get an expression for $\delta\alpha_i$

$$\delta\alpha_i = \tau \sum_j (\langle \hat{O}_i \rangle \left\langle \hat{O}_j^\dagger \right\rangle + \left\langle \hat{O}_j^\dagger \hat{O}_i \right\rangle)^{-1} (\langle \hat{H} \rangle \left\langle \hat{O}_j^\dagger \right\rangle - \left\langle \hat{O}_j^\dagger \hat{H} \right\rangle) \tag{5.61}$$

which is easier to see in vector notation with matrix S with elements $S_{ij} = \langle \hat{O}_i \rangle \left\langle \hat{O}_j^\dagger \right\rangle + \left\langle \hat{O}_j^\dagger \hat{O}_i \right\rangle$, $\vec{\delta\alpha} = (\delta\alpha_1, \delta\alpha_2, ..., \delta\alpha_n)$ and vector $\vec{f}$ with elements $\vec{f}_j = \langle \hat{H} \rangle \left\langle \hat{O}_j^\dagger \right\rangle - \left\langle \hat{O}_j^\dagger \hat{H} \right\rangle$ the Equation becomes

$$\vec{\delta\alpha} = S^{-1} \vec{f} \tag{5.62}$$

5.4 Literature Review

There are extensive and excellent works on the role of machine learning in quantum chemistry [117, 154]. The aim in this Section is to connect the dots of incumbent methods for modelling wave functions, described in previous sections, and relatively new neural network based methods.

Methods involving supervised learning are not explored, as these do not involve solving the Schrödinger equation, at least not directly. In general, these methods attempt to predict properties of systems given labelled data about other systems, either from experiment or data from *ab initio* methods. A field in it's own right, it is not discussed here as the techniques and goal (predicting the behaviour of new systems given data on old systems) are distinct from our

own (modelling the wave function of a given system). I refer to other work for information on this field [73].

Machine learning methods, in the purest sense of algorithms that iteratively improve given more data, have been computational chemistry tools for some time [90]. Recently, machine learning has become almost synonymous with neural networks in 'deep learning', and this holds true in computational chemistry, where these relatively novel methods are breaking ground in diverse areas of research.

Using neural networks to solve *ab initio* computational chemistry problems, at a high level, involves creating a physics-based model of a system and representing some or all of that physics model with neural networks. Widely recognised as the first attempt, in Reference [51] the authors use a Boltzmann machine to model ground states of spin systems. In the following years Boltzmann machines remained popular in this area, despite the incredible successes of feedforward neural networks trained via backpropagation. Boltzmann machines have been applied extensively to learn wave function of static systems [69, 191, 234, 252, 253, 294], understand the capacity of models for modelling many-body quantum systems [149, 174, 234]. Although, they are widely regarded in the classical machine learning community as the more expensive and less effective method [76].

More recently, research interest has turned to state-of-the-art methods. For example, researchers have developed new algorithms [224], sampling processes [291], physics-based approaches [108], applied to open quantum systems [184] and successful methods that support the deep learning mantra 'bigger is better' [204]. Reviews in these areas include [117, 154].

Other models and systems explored by researchers for modelling quantum states include feedforward neural networks [49, 128, 288, 291], variational autoencoders [217], and more recently convolutional neural networks (specifically for frustrated systems) [64], convolutional [63, 64, 149, 151, 177, 253], variational autoregressive [53, 287], and for the simulation of open quantum systems [252].

The specific model we explore in this Thesis is an adaption of the Slater-Jastrow-backflow wave function where backflow has previously been modelled by neural networks [157, 224, 275].

5.5 Homogeneous Electron Gas

In this section I outline a brief example of combining the previously discussed elements into a workable framework for solving the time-independent Schrödinger equation for a fermionic system.

The homogeneous electron gas [57, 74] is a system of electrons moving in a field. We wish to find the ground state wave function of the system. They can be described as a Fermi fluid; a Fermi fluid is a liquid of particles having Fermi-Dirac statistics. Fermi-Dirac statistics describes the distribution of population over states of systems of many identical fermionic particles obeying

the Pauli exclusion principle. Another way of saying this is particles with half-integer spin at thermodynamic equilibrium.

Taking the Schrödinger equation for some parameterized Ansatz $\psi(X;\theta)$, where we drop the θ dependence,

$$\hat{H}\psi(X) = E\psi(X). \tag{5.63}$$

The Hamiltonian of the system is

$$\hat{H} = -\frac{1}{r_s^2}\sum_{i=1}^{N}\nabla_i^2 + \frac{2}{r_s}\sum_{i<j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + c, \tag{5.64}$$

where r_s is the density parameter and c is a constant potential factor resulting from a homogeneous positive field around the electrons. It is disregarded in the local energy computations, Equation (5.46), as its only effect is to shift the entire eigenspectrum up, which is trivial to recover. The density parameter is important for determining the ground state properties of the free electron gas and is given by

$$r_s = n_e/(a/a_0) \tag{5.65}$$

where a_0 is the Bohr radius and a is the mean separation of the electrons and n_e is the number of electrons. In a 3D model we can compute the volume v, given a, of the sphere occupied by a single electron

$$\frac{4}{3}\pi a^3 = v, \tag{5.66}$$

which given some n_e determines the total volume of the box. The length of a side of the box l can be computed $l^3 = v$. The size of the box and the number of electrons, which determine the density parameter r_s, are hyperparameters of the experiment. The density determines the behaviour of the system (i.e. the phase of the electron gas), for example the Wigner crystal is the solid crystalline phase of electrons [279], and the size of the box determines the accuracy of the model of the system; in the limit $n_\mathrm{e} \to \infty$ and $l \to \infty$, the thermodynamic limit, the simulation is exact. Now we can decide how to model our wave function. For this example, we choose a Slater-Jastrow-backflow wave function of the form

$$\psi(X) = \mathrm{e}^{J(X')}\sum_i \mathbf{\Phi}_i(X') \tag{5.67}$$

X' are the set of backflow system coordinates $X' = \mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}_{n_\mathrm{r}}$, where the coordinates are shifted by the backflow transform

$$\mathbf{r}'_i = \mathbf{r}_i + \sum_{i \neq j} \eta(r_{ij})(\mathbf{r}_i - \mathbf{r}_j). \tag{5.68}$$

The Slater determinants in Equation (5.67) can be decomposed into a product of spin up and spin down determinants,

$$\psi(X) = \mathrm{e}^{J(X')} \sum_{i=1}^{k} \det\left[\mathbf{\Phi}_i^{\uparrow}(X')\right] \det\left[\mathbf{\Phi}_i^{\downarrow}(X')\right]. \tag{5.69}$$

This representation forces off-block-diagonal elements of the full determinant matrix Φ to zero:

$$\begin{aligned} \Phi_{ij} &= 0 \text{ if } (i \in \{1,\ldots,n_\uparrow\} \wedge j \in \{n_\uparrow + 1,\ldots,n\}) \\ &\quad \vee (i \in \{n_\uparrow + 1,\ldots,n\} \wedge j \in \{1,\ldots,n_\uparrow\}) \\ &= 0 \text{ if } (i \leq n_\uparrow \wedge j > n_\uparrow) \vee (i > n_\uparrow \wedge j \leq n_\uparrow). \end{aligned} \tag{5.70}$$

The Jastrow factor and backflow functions can be modelled by a an arbitrary function, such as a neural networks parameterized some set of parameters θ, provided the $\mathbf{\Phi}(X';\theta)$ are antisymmetric and $J(X';\theta)$ is symmetric and satisfies the boundary conditions of the cell and the electron-electron cusp conditions at coalescence. The final parameterized Ansatz in this construction is

$$\psi(X;\theta) = J(X';\theta) \sum_{i=1}^{k} \det\left[\mathbf{\Phi}_i^{\uparrow}(X';\theta)\right] \det\left[\mathbf{\Phi}_i^{\downarrow}(X';\theta)\right] \tag{5.71}$$

where gradients can be computed via Metropolis Hastings sampling, Algorithm 2, and the parameter updates by optimization, Section 5.3.1.

CHAPTER 6

FERMIONIC NEURAL NETWORKS AND KRONECKER FACTORED APPROXIMATE CURVATURE

... the farther down you go, the bigger and scalier the turtles get, with sharper beaks.
— Lev Grossman, *The Magicians*

Recently developed neural network-based *ab-initio* solutions (Pfau et. al arxiv:1909.02487v2) for finding ground states of fermionic systems can generate state-of-the-art results on a broad class of systems. In this work, we improve the results for this Ansatz with Diffusion Monte Carlo. Additionally, we introduce several modifications to the network (FermiNet) and optimization method (Kronecker Factored Approximate Curvature) that reduce the number of required resources while maintaining or improving the modelling performance. In terms of the model, we remove redundant computations and alter the way data is handled in the permutation equivariant function. The Diffusion Monte Carlo results exceed or match state-of-the-art performance for all systems investigated: atomic systems Be-Ne, and the carbon cation C^+.

6.1 Introduction

Neural networks, in recent years, have provided an alternative computational paradigm for solving electronic structure problems which includes applications in directly solving the time-independent Schrödinger equation to find approximate ground states of atoms and molecules [267]. Standard quantum chemistry methods [249], such as coupled-cluster [139], full configuration interaction [221], VMC [172], and DMC [257] have been used in conjunction with neural network methods in different ways, such as the introduction of new Ansätze [51, 108, 204, 232, 233, 255] or providing rich datasets for the prediction of properties of previously untested systems and their dynamics in supervised learning frameworks [25, 66, 90, 292].

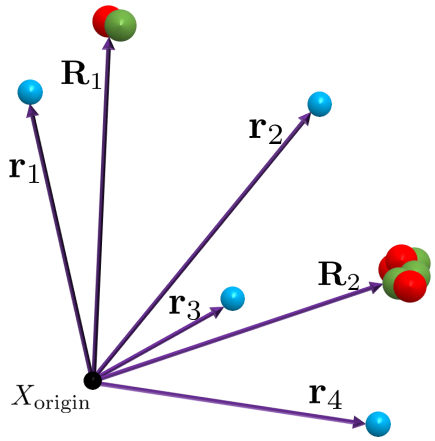


Figure 6.1: The system of atoms (protons/neutrons red/green) and electrons (blue). The set of position vectors of a system of atoms, $\mathbf{R}_i$ and electrons $\mathbf{r}_i$. X_{origin} is the origin (black) of the coordinate system. In all cases explored here we considered single atom systems and the origin was set to the nucleus position. However, the choice of origin is completely arbitrary as FermiNet is invariant to translations.

Notable examples of new Ansätze are neural networks such as PauliNet [108], Boltzmann machines [51] and FermiNet [204]. Even though these techniques are still in the early stages of development, neural networks for fermionic systems has been widely studied over the past few years, and some of these techniques are capable of producing state-of-the-art results for electronic structure simulations. We have particular interest in finding good approximations to ground states of fermionic Hamiltonians [108] and the precision and accuracy of FermiNet [204].

In this paper we modify an existing framework, the FermiNet [204], by changing how data is handled in the network and removing redundant elements. We apply the wave function optimization algorithm VMC whilst altering some aspects of the KFAC optimization, and then running the DMC algorithm to improve the wave function further. The use of DMC is standard practice in many Quantum Monte Carlo (QMC) codes, however, the wave functions used in virtually all packages prior to this new wave of neural network approaches are using a wave function engine consisting of Slater/Jastrow/Multi-determinant/Backflow wave functions, which generally are less accurate than the FermiNet Ansatz.

The QMC we described above is generally run with standard VMC and then DMC with the fixed node approximation (nuclei are fixed in place) in the continuum. That is to say we solve the Schrödinger equation

$$\hat{H}\psi(X) = E\psi(X), \tag{6.1}$$

where X defines the system configuration, Figure 6.1, $\hat{H}$ is the Hamiltonian operator and E is the energy of the wave function $\psi(X)$.

The Chapter is organized as follows: Section 6.2 introduces the problem to be solved (i.e. the time-independent Schrödinger equation for fermionic systems) and all the relevant background including VMC, KFAC, DMC and a sketch of the FermiNet; Section 6.3 describes the particular methods used in this work including detailed descriptions of the algorithms; The results are described and discussed in Section 6.4; and finally we conclude our findings in Section 6.5.

Contributions

- Introduced changes to the FermiNet implementation, removing the diagonal elements of the pairwise terms and altering how the data is handled in the permutation equivariant function, resulting in improvements in efficiency, described in Section 6.2.1
- First known application of Diffusion Monte Carlo with a neural network Ansatz
- State-of-the-art results on all systems explored (Be-Ne, and C^+)

This Ansatz is functionally identical to previous work. We refer to the model and the associated optimization methods described as FermiNet* in order to distinguish from previous work that is referred to as FermiNet.

At a high level, the FermiNet is a function mapping configurations (particle positions) to amplitudes. This function needs to be anti-symmetric under the exchange of same spin electrons, which enforces some structure to the function. The anti-symmetry is implemented with a determinant operation on orbitals (exponentially decaying functions), where the determinant is anti-symmetric under the exchange of rows/columns, and permutation equivariant functions before the determinant generating the orbitals. These orbitals are functions of all electron positions which allows the wave function to capture correlations between particles.

6.2 Solving the Schrödinger equation for fermionic systems

6.2.1 Fermionic Neural Network Ansatz

The Fermionic Neural Network (FermiNet) is a neural network designed specifically for the task of representing the wave function, in continuous Euclidean space, of the Schrödinger equation for a fermionic Hamiltonian. In this section we describe the original model, found in Reference [204], and in Section 6.3.1 we describe FermiNet*, which is functionally identical but better performing (faster) than the original.

At a high level, the FermiNet consists of

I learnable single electron features (single streams),

II learnable electron-electron interaction features (pairwise streams),

III permutation equivariant operations (EQV),

IV and multi-electron orbitals.

To a lesser extent the implementation and optimization details are necessary to the performance and usage. As such they are characteristic of the FermiNet implementation:

VI KFAC optimization;

VII and stable log-domain computation of the amplitudes, first order derivatives and second order derivatives.

Next, in this background, we give an overview of the FermiNet model, describing I-IV. Then we detail KFAC, describing VI. Details on VII, including the LogSumExp trick and derivations of the derivatives of the determinant, can be found in the appendix of Reference [204] or understood from other references [38].

Ansatz

There are two sets of streams in the network referred to as the single and pairwise streams. These streams contain the data corresponding to single, $\mathbf{h}_i^{l\alpha}$, and pairwise, $\mathbf{h}_{ij}^{l\alpha\beta}$, electron input features, respectively. These variables are indexed by l, the layer of the network, and i and j, the electron indexes. α is the spin of electron i and β is the spin of electron j.

The inputs, computed from the system X, to the network are indexed by $l = 0$ and computed as

$$\mathbf{h}_i^{0\alpha} = (\mathbf{r}_i - \mathbf{R}_0, \|\mathbf{r}_i - \mathbf{R}_0\|, \mathbf{r}_i - \mathbf{R}_1, \|\mathbf{r}_i - \mathbf{R}_1\|, ..., \mathbf{r}_i - \mathbf{R}_n, \|\mathbf{r}_i - \mathbf{R}_n\|), \tag{6.2}$$

$$\mathbf{h}_{ij}^{0\alpha\beta} = (\mathbf{r}_i - \mathbf{r}_j, \left\|\mathbf{r}_i - \mathbf{r}_j\right\|). \tag{6.3}$$

$\mathbf{r}_i$ and $\mathbf{R}_j$ are the electron and atom position vectors, as shown in Figure 6.1, and $\|\cdot\|$ is the Euclidean norm.

The data from the streams at each layer are transformed by a permutation equivariant function. The permutation equivariant function maintains the anti-symmetry (required by fermionic systems) of the Ansatz and generates multi-electron orbitals with demonstrably good modelling capacity. This function outputs one streams of data $\mathbf{f}_i^{l\alpha}$:

$$\mathbf{f}_i^{l\alpha} = \left(\mathbf{h}^{l\alpha}, \frac{1}{n_\uparrow}\sum_{\beta\neq\downarrow}\mathbf{h}_{ij}^{l\alpha\beta}, \frac{1}{n_\downarrow}\sum_{\beta\neq\uparrow}\mathbf{h}_{ij}^{l\alpha\beta}, \frac{1}{n_\uparrow}\sum_{\alpha\neq\downarrow}\mathbf{h}_i^{l\alpha}, \frac{1}{n_\downarrow}\sum_{\alpha\neq\uparrow}\mathbf{h}_i^{l\alpha}\right) \tag{6.4}$$

The data at each layer l are computed

$$\begin{aligned}\mathbf{h}_i^{l\alpha} &= \tanh\left(\mathbf{W}_l \mathbf{f}_i^{l\alpha} + \mathbf{b}^l\right) + \mathbf{h}_i^{(l-1)\alpha} \\ \mathbf{h}_{ij}^{l\alpha\beta} &= \tanh\left(\mathbf{V}_l \mathbf{h}_{ij}^{l\alpha\beta} + \mathbf{c}_l\right) + \mathbf{h}_{ij}^{(l-1)\alpha\beta}\end{aligned} \tag{6.5}$$

where in a layer l, $\mathbf{W}_l$, and $\mathbf{V}_l$ are weights, $\mathbf{b}_l$ and $\mathbf{c}_l$ are the biases. Residual connections are added to all layers where $\dim(\mathbf{h}^l) = \dim(\mathbf{h}^{(l-1)})$.

There are n_l of these parameterized layers. The outputs $\mathbf{f}_i^{L\alpha}$ are split into spin dependent data blocks. There is a linear transformation to map $\mathbf{h}_i^{L\alpha}$ to a scalars which are coefficients of the exponentials in the orbitals. Multiple determinants are generated in this way, indexed by k, which contribute to the modelling capacity of the network and the elements of the determinants are the product of the coefficient computed by the FermiNet and the envelopes

$$\phi_{ij}^{\alpha k} = (\mathbf{w}_{li}^{\alpha k} \mathbf{h}_j^{l\alpha} + d_{li}^{\alpha k}) \times \sum_m \pi_{im}^{\alpha k} \exp\left(-|\Sigma_{im}^{\alpha k}(\mathbf{r}_j^{\alpha} - \mathbf{R}_m)|\right) \tag{6.6}$$

$\mathbf{\Sigma}_{im}^{\alpha k}$ control the anisotropic (direction dependent) behaviour of the envelope whereas the inverse exponential ensures the wave function decays to zero when the electrons are large distances from the nuclei.

The determinants are constructed

$$\det\left[\Phi^{\alpha k}\right] = \begin{vmatrix} \phi_{00}^{\alpha k} & \cdots & \phi_{0n}^{\alpha k} \\ \vdots & & \vdots \\ \phi_{n0}^{\alpha k} & \cdots & \phi_{nn}^{\alpha k} \end{vmatrix}. \tag{6.7}$$

and the amplitudes are computed from these

$$\psi(X) = \sum_k \omega_k \det\left[\Phi^{\uparrow k}\right] \det\left[\Phi^{\downarrow k}\right]. \tag{6.8}$$

Equation (6.8) is a representation of the full determinant as a product of spin up and down determinants. This representation forces off-block-diagonal elements of the full determinant Φ to zero, when

$$\begin{aligned}&(i \in \{1,...,n_\uparrow\} \wedge j \in \{n_\uparrow + 1,...,n\}) \vee (i \in \{n_\uparrow + 1,...,n\} \wedge j \in \{1,...,n_\uparrow\}) \\ &= (i \leq n_\uparrow \wedge j > n_\uparrow) \vee (i > n_\uparrow \wedge j \leq n_\uparrow),\end{aligned} \tag{6.9}$$

where i and j refer to the orbital index. Finally, the sign is split from the the amplitudes and the network outputs the amplitudes in the log-domain for numerical stability

$$\log|\psi(X)| = \log\left|\sum_k \omega_k \det\left[\Phi^{\uparrow k}\right]\det\left[\Phi^{\downarrow k}\right]\right| \tag{6.10}$$

Other work [245], removes the weights ω_k in Equation (6.8), as they are functionally redundant, and replaces the anisotropic decay parameters $\mathbf{\Sigma}_{im}^{\alpha k}$ in Equation (6.6) with a single parameter, restricting the orbitals to isotropic decay. Surprisingly, the authors note this does not seem to result in a decrease in modelling accuracy but does lead to significant gains in speed. These improvements are further discussed in Section 6.4.

Kronecker Factored Approximate Curvature

Natural gradient descent [9], is an algorithm for computing the natural gradients of a parameterized function. They are

$$\Delta = F^{-1}\mathrm{vec}(\Delta\mathscr{L}) \tag{6.11}$$

where Δ are the natural gradients, $\Delta\mathscr{L}$ are the gradients of the function parameters and F is the Fisher Information Matrix (FIM)

$$F = \mathbb{E}_X\left[\frac{d\log p_\theta(X)}{d\theta}^T \frac{d\log p_\theta(X)}{d\theta}\right] \tag{6.12}$$

and g are the gradients computed in Riemmanian space. KFAC is an algorithm which approximates the natural gradients of a neural network. There are a series of approximations and methods associated with this algorithm, which are often applied in a case dependent way (as we will see in this work). The bottleneck of Natural Gradient Descent is the cost of inverting the FIM, which for an $n \times n$ matrix is $\sim O(n^3)$. This is prohibitive for neural networks with even a few thousand parameters. KFAC reduces this burden whilst still accurately (enough) modelling the Fisher such that useful approximate natural gradients can be found.

Given that the inverse of a Kronecker product is the Kronecker product of the inverses

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{6.13}$$

the authors break down the FIM into blocks, extract the most important blocks, and approximate these blocks in a form that allows the identity of inverting Kronecker products, Equation (6.13), to be used.

Expressing the FIM as blocks it is natural to use the structure of the neural network to dictate the layout of the blocks:

$$F = \begin{pmatrix} \mathbb{E}[D\theta^{(0)} D\theta^{(0)T}] & \cdots & \mathbb{E}[D\theta^{(0)} D\theta^{(L)T}] \\ \vdots & \ddots & \vdots \\ \mathbb{E}[D\theta^{(0)} D\theta^{(L)T}] & \cdots & \mathbb{E}[D\theta^{(L)} D\theta^{(L)}] \end{pmatrix} \tag{6.14}$$

where

$$D\theta^{(i)} = \frac{d \log p_\theta(X)}{d\theta^{(i)}} \tag{6.15}$$

the $^{(i)}$ superscript denotes the parameters in layer i of the network.

Removing all but the diagonal blocks we have

$$\tilde{F} = \begin{pmatrix} \mathbb{E}[D\theta^{(0)} D\theta^{(0)T}] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbb{E}[D\theta^{(L)} D\theta^{(L)}] \end{pmatrix} \tag{6.16}$$

Each of these blocks can be reformulated as the Kronecker product of the outer product of the activtations and sensitivities, for example layer 0,

$$\begin{aligned} F^{(0)} &= \mathbb{E}_X[D\theta^{(0)} D\theta^{(0)T}] \\ &= \mathbb{E}_X[(\mathbf{a}_0 \otimes \mathbf{s}_0)(\mathbf{a}_0 \otimes \mathbf{s}_0)^T] \\ &= \mathbb{E}_X[\mathbf{a}_0 \mathbf{a}_0^T \otimes \mathbf{s}_0 \mathbf{s}_0^T] \end{aligned} \tag{6.17}$$

and finally assuming that the expectation of Kronecker product is approximately equal to the Kronecker product of the expectations, which has no name so we simply call the KFAC approximation here, though technically more accurately is independence of the covariances of the activations and covariances of the sensitivities,

$$\begin{aligned} F^{(l)} &\overset{\text{KFAC}}{\approx} \mathbb{E}_X[\mathbf{a}_l^T \mathbf{a}_l] \otimes \mathbb{E}_X[\mathbf{s}_l^T \mathbf{s}_l] \\ &= A_l \otimes S_l. \end{aligned} \tag{6.18}$$

In practise, it is common to smooth the approximation to the Fisher in time. At any iteration in the optimization the quality of the approximation is limited by the number of samples in a batch. This can be unstable. Instead, the approximation is smoothed by replacing the left and right Fisher factors with their moving averages. There are different methods for implementing this exponentially decaying average of the history, in this work it is computed

$$\bar{A}_{lt} = (\bar{A}_{l(t-1)} + \kappa A_{lt})/\varkappa_t \tag{6.19}$$

where $\varkappa_t = \varkappa_{(t-1)} + \kappa$ is defined as the total weight and $\varkappa_0 = 0$, $\bar{A}_{l0}$ is a matrix of zeros and A_{lt} is the instantaneous covariances computed from Equation (6.18). The right Fisher factor is computed in the same way. It is important to be careful here that the contribution to the exponentially decaying average of the moving covariances from the zeroth term is zero, otherwise the following computed averages will be zero biased.

The approximate natural gradients of each Fisher block can be computed as Equation (6.11),

$$\tilde{\delta}_l = \tilde{F}^{(l)-1}\mathrm{vec}(\Delta\mathscr{L}_l) \tag{6.20}$$

We use $\tilde{\delta}$ to indicate approximate natural gradients throughout and $\mathrm{vec}(X)$ is the vectorization of a matrix X.

6.2.2 Damping

This form of KFAC is unstable for two reasons: The approximate natural gradients can be large and the Fisher blocks are potentially singular (thus the inversion can't be performed).

Damping is a tool for shifting the eigenspectrum of a matrix (such that the matrix is non-singular) and also restricting the 'trust region' of the updates. The inverse Fisher described in Equation (6.20) is adjusted, typically in Tikhonov damping,

$$\bar{\Delta}_l = (\tilde{F}^{(l)} + \lambda)^{-1}\mathrm{vec}(\Delta\mathscr{L}_l) \tag{6.21}$$

This can no longer be decomposed as Equation (6.13). An adapted technique, Factored Tikhonov Damping described in detail in Reference [166], is used which approximates the true damping

$$(\tilde{F}^{(l)} + \lambda)^{-1} \overset{\mathrm{FT}}{\approx} (A + \sqrt{\pi\lambda}) \otimes (S + \sqrt{\lambda/\pi}) \tag{6.22}$$

where

$$\pi_l = \frac{\mathrm{Tr}[A_l]\dim(S_l)}{\mathrm{Tr}[S_l]\dim(A_l)} \tag{6.23}$$

This method is derived from minimizing the residual from the expansion of damped Fisher block into Kronecker factors.

6.2.3 Centering

All approximations to the Fisher block have thus far been computed via the derivatives of the log-likelihood of the normalized distribution $p(X)$ with respect to the weights. However, we compute $\mathcal{O} = \frac{d \log|\psi(X)|}{d\theta}$ which are the derivatives of the log unnormalized wave function $\psi(X)$. It can be shown, for example Appendix C of Reference [204] that elements in the FIM can be expressed

$$F_{ij} \propto \mathbb{E}_X \left[(\mathcal{O}_i - \mathbb{E}_X[\mathcal{O}_i])(\mathcal{O}_j - \mathbb{E}_X[\mathcal{O}_j]) \right] \tag{6.24}$$

$$= \frac{1}{4} \mathbb{E}_X \left[\frac{d \log p(X)}{d\theta_i} \frac{d \log p(X)}{d\theta_j} \right] \tag{6.25}$$

Where the derivatives of the log wave function have been centered by $\mathbb{E}_X[\mathcal{O}]$. However, in tests we found that the performance was more dependent on the hyperparameters used in KFAC than on the centering. There are many approximations affecting the quality of the approximation to the FIM and the interplay between the optimization and this approximation. We ignored the centering to simplify the implementation but outline the method here of approximating this centering for readers who are interested.

$$\begin{aligned} F &\propto \mathbb{E}_X \left[(\mathcal{O} - \mathbb{E}_X[\mathcal{O}])^T (\mathcal{O} - \mathbb{E}_X[\mathcal{O}]) \right] \\ &= \mathbb{E}_X \left[\mathcal{O}^T \mathcal{O} - \mathbb{E}_X[\mathcal{O}]^T \mathcal{O} - \mathcal{O}^T \mathbb{E}_X[\mathcal{O}] \right. \\ &\quad \left. - \mathbb{E}_X[\mathcal{O}]^T \mathbb{E}_X[\mathcal{O}] \right] \\ &\overset{\text{LINEAR}}{=} \mathbb{E}_X[\mathcal{O}^T \mathcal{O}] - \mathbb{E}_X[\mathcal{O}]^T \mathbb{E}_X[\mathcal{O}] \end{aligned} \tag{6.26}$$

(6.27)

The second term can be approximated by assuming Independent Activations and Sensitivities (IAD):

$$\mathbb{E}_X[\mathcal{O}]^T \mathbb{E}_X[\mathcal{O}] = \mathbb{E}_X[\mathbf{a} \otimes \mathbf{s}]^T \mathbb{E}_X[\mathbf{a} \otimes \mathbf{s}] \tag{6.28}$$

$$\overset{\text{IAD}}{\approx} (\mathbb{E}_X[\mathbf{a}] \otimes \mathbb{E}_X[\mathbf{s}])^T (\mathbb{E}_X[\mathbf{a}] \otimes \mathbb{E}_X[\mathbf{s}]) \tag{6.29}$$

$$= \mathbb{E}_X[\mathbf{a}]^T \mathbb{E}_X[\mathbf{a}] \otimes \mathbb{E}_X[\mathbf{s}]^T \mathbb{E}_X[\mathbf{s}] \tag{6.30}$$

$$= A' \otimes S'. \tag{6.31}$$

Overall, the centering outlined in Equation (6.26) can be approximated by mean centering the activations and sensitivities

$$\bar{\mathbf{a}} = \mathbf{a} - \mathbb{E}_X[\mathbf{a}] \tag{6.32}$$

$$\bar{\mathbf{s}} = \mathbf{a} - \mathbb{E}_X[\mathbf{s}] \tag{6.33}$$

or alternately maintaining a moving average of A' and S' and subtracting those directly from $\bar{A}$ and $\bar{S}$ in Equation (6.19). Both these methods result in the same residual, which can be computed by expanding either expression for the resultant approximate FIM.

6.2.4 This work

The optimization algorithm used here is a variant approximate natural gradient descent method known as KFAC. Roughly speaking, updates (the approximate natural gradients) $\tilde{\delta}_l$ for layer l are computed as

$$\begin{aligned} \tilde{\delta}_l &= \tilde{F}_l^{-1}\text{vec}(\Delta_l \mathscr{L}) \\ &= \bar{A}_l^{-1}\Delta_l \mathscr{L} \bar{S}_l^{-1}, \end{aligned} \tag{6.34}$$

where $\tilde{F}_l^{-1}$ is the approximate inverse Fisher block, $\Delta_l \mathscr{L}$ are the gradients corresponding to weights in layer l, and $\bar{A}_l$ and $\bar{S}_l$ are the moving covariances of the left and right Fisher factors, respectively, discussed in literature [22, 96, 165, 166]. The term Fisher block is used to indicate the elements of the FIM corresponding to a given layer in the network. In Reference [204] the authors use a reduced version of the full KFAC algorithm. They do not use adaptive damping or adaptive learning rates which are somewhat characteristic of the original algorithm [203] and considered important in the literature [166].

The KFAC variant used in this work is closer to an adaption named Kronecker Factors for Convolution [96] because the weights in most of the layers are reused. For example, the single stream weights are used n_e times, where n_e is the number of electrons in the system. There are different approaches to approximating the effect this has on the FIM and here we use the simpler and more efficient approximation developed in Reference [22]

$$\tilde{F}_l = |T|^2 \mathbb{E}_X\left[\mathbb{E}_i[\mathbf{a}_{li}]\mathbb{E}_i[\mathbf{a}_{li}]^T\right] \otimes \mathbb{E}_X\left[\mathbb{E}_i[\mathbf{s}_{li}]\mathbb{E}_i[\mathbf{s}_{li}]^T\right] \tag{6.35}$$

where ^{T} indicates the transpose, $\mathbf{a}_{li}$ are the activations of spatial location (the index of data which are operated on by the same parameters) i at layer l and $\mathbf{s}_{li}$ are the sensitivities of the pre-activations $\mathbf{z}_{li}$ of layer l

$$\mathbf{s}_{li} = \frac{d\log|\psi(X)|}{d\mathbf{z}_{li}}. \tag{6.36}$$

See Figure 6.2 for a sketch of how these variables are related. $\mathbb{E}_i$ is the expectation taken over all spatial locations

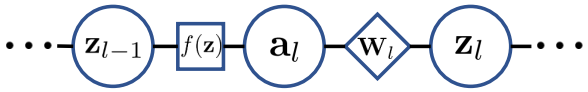


Figure 6.2: Sketch of the relationship between $\mathbf{a}$ and $\mathbf{z}$. For some layer l, $\mathbf{z}_l$ are the pre-activations, $f(\mathbf{z})$ is an activation function, $\mathbf{a}_l$ are the activations and $\mathbf{w}_l$ are the weights. Data variables are in circles, functions in square and network parameters in diamond.

$$\mathbb{E}_i[\mathbf{a}_{li}] = \frac{1}{|T|}\sum_i \mathbf{a}_{li} \tag{6.37}$$

$$\mathbb{E}_i[\mathbf{s}_{li}] = \frac{1}{|T|}\sum_i \mathbf{s}_{li} \tag{6.38}$$

where T is the set of spatial locations and $|T|$ is its cardinality.

6.3 Methods

6.3.1 FermiNet*

Name	symbol	value
Single Stream hidden units	n_s	256
Pairwise Stream hidden units	n_p	32
Split Stream hidden units	n_{ss}	256
Determinants	n_k	16
Layers	n_l	4

Table 6.1: Model hyperparameters.

We change the network slightly, decreasing the resources (defined as the number of operations) required by the original model. We refer to this implementation of the model (and distinct optimization) as FermiNet* in order to distinguish from other work, Reference [204], which is referred to as FermiNet. For both implementations of the model we compute the total number of operations of the implementations and compare the walltime.

The number of operations n_{ops} is computed as

$$n_{\text{ops}} = \sum_l n^l_{\text{uses}} \times d^l_{\text{out}} \times (d^l_{\text{in}} + (d^l_{\text{in}} - 1)) \tag{6.39}$$

where l is index running over all layers performing matrix multiplications and n^l_{uses}, d^l_{out} and d^l_{in} are the number of times those weights are used, the input dimension and the output dimension, respectively. To clarify, $d_{\text{out}} \times (d_{\text{in}} + (d_{\text{in}} - 1))$ is the cost of one matrix-vector product in

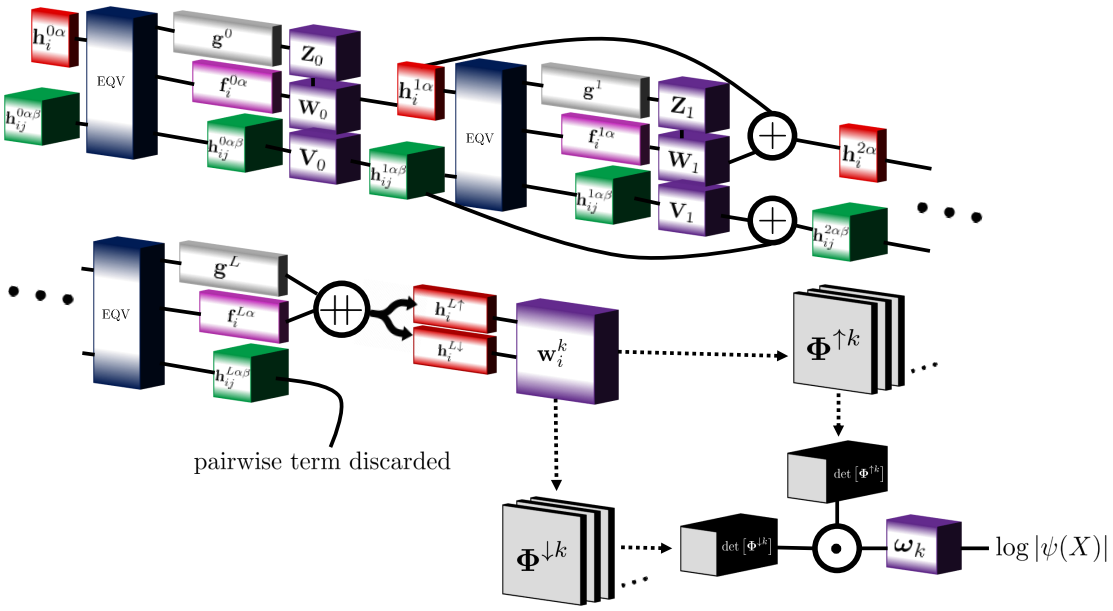


Figure 6.3: Overview of FermiNet. The system description X is used to compute the single stream and pairwise stream input feature tensors $\mathbf{h}_i^{0\alpha}$ and $\mathbf{h}_i^{0\alpha\beta}$, respectively. These are passed to the permutation equivariant function (EQV), Figure 6.4. Linear layers are applied to the resulting tensors with tanh activations. Outputs of the Split Stream and Single Stream matrix multiplications are combined before a tanh activation in the Single Stream layer. These layers are repeated 4 times. After the final permutation equivariant function, the Split Stream and Single Stream outputs are concatenated (++) and Pairwise Stream data discarded. The concatenated tensor is passed through a final spin dependent linear transformation to spin-up and spin-down determinants. The final layer is a custom computation of the determinants which involves stable first- and second-order derivatives and the LogSumExp trick.

terms of the number of operations (multiplication or addition). This equation only computes the contributions from the linear layers because these are the only layers where the implementations differ and ignores other operations such as computing the determinant, which is the dominant operation in the complexity.

It is possible to completely remove elements from the pairwise streams tensor with no effect to the modelling capacity or performance of the network. The diagonal elements of the pairwise tensor $\mathbf{h}_{ij}^{l\alpha\beta}$ where $i = j$ are redundant. The inputs computed from Equation (6.3) are zero, have zero contribution to computation of the energy (and therefore zero contribution to the computation of the gradients). Though this only results in a small reduction in resource, Figure 6.5(b), we find a per iteration walltime reduction of $\sim 5-10\%$. The effect is more noticeable at larger system sizes.

The permutation equivariant function is changed to decrease the number of operations, Figure 6.4. This new function outputs two streams of data, $\mathbf{f}_i^{l\alpha}$ and $\mathbf{g}^l$:

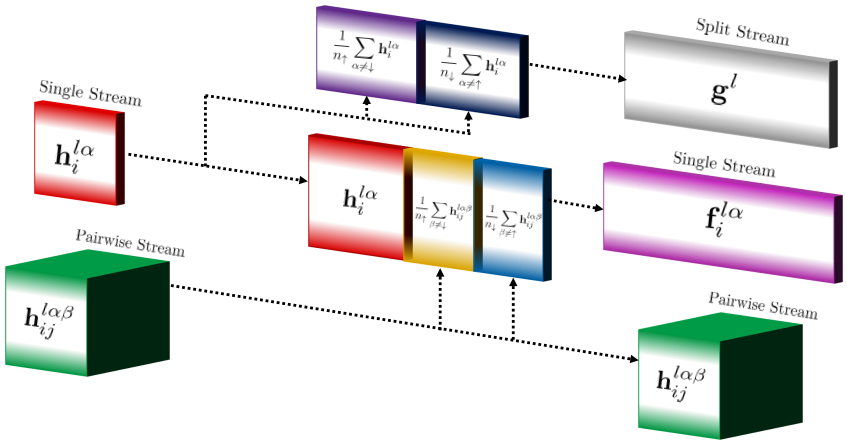


Figure 6.4: Data from the single and pairwise streams are combined in this operation via Equations (6.40) & (6.41). The pairwise stream data remains unchanged.

$$\mathbf{f}_i^{l\alpha} = \left(\mathbf{h}^{l\alpha}, \frac{1}{n_\uparrow}\sum_{\beta\neq\downarrow}\mathbf{h}_{ij}^{l\alpha\beta}, \frac{1}{n_\downarrow}\sum_{\beta\neq\uparrow}\mathbf{h}_{ij}^{l\alpha\beta}\right) \tag{6.40}$$

$$\mathbf{g}^l = \left(\frac{1}{n_\uparrow}\sum_{\alpha\neq\downarrow}\mathbf{h}_i^{l\alpha}, \frac{1}{n_\downarrow}\sum_{\alpha\neq\uparrow}\mathbf{h}_i^{l\alpha}\right). \tag{6.41}$$

This implementation requires less resources than the original, as the mean over spin terms in the layer, $\mathbf{g}^l$, are only operated on once, instead of n_e times, in the case that these data are not split and $\mathbf{g}^l$ appears in all single electron streams.

The data at each layer l are computed

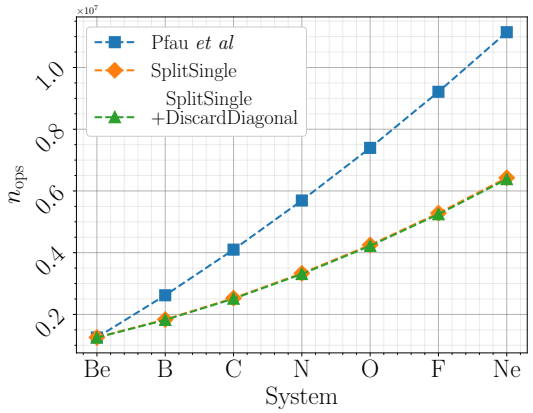
$$\mathbf{h}_i^{l\alpha} = \tanh\left(\mathbf{W}_l\mathbf{f}_i^{l\alpha} + \mathbf{Z}_l\mathbf{g}^{l-1} + \mathbf{b}^l\right) + \mathbf{h}_i^{(l-1)\alpha} \tag{6.42}$$

$$\mathbf{h}_{ij}^{l\alpha\beta} = \tanh\left(\mathbf{V}_l\mathbf{h}_{ij}^{l\alpha\beta} + \mathbf{c}_l\right) + \mathbf{h}_{ij}^{(l-1)\alpha\beta} \tag{6.43}$$

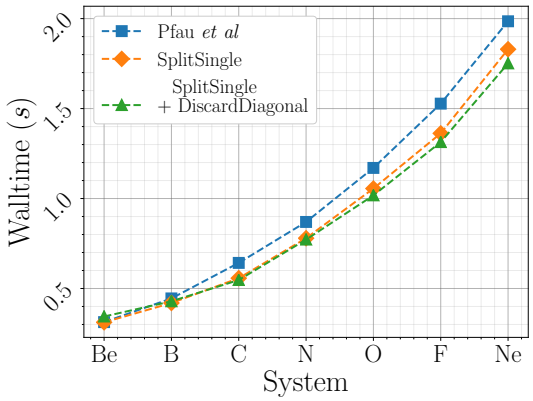
where in a layer l, $\mathbf{W}_l$, $\mathbf{Z}_l$ and $\mathbf{V}_l$ are weights, $\mathbf{b}_l$ and $\mathbf{c}_l$ are the biases. As before, residual connections are added to all layers where $\dim(\mathbf{h}^l) = \dim(\mathbf{h}^{(l-1)})$. The outputs $\mathbf{f}_i^{L\alpha}$ and $\mathbf{g}^L$ are concatenated, $\mathbf{h}_i^{L\alpha} = (\mathbf{f}_i^{L\alpha}, \mathbf{g}^L)$, and split into two spin dependent data blocks.

This is an alternate but equivalent representation of the permutation equivariant function outlined in Reference [204]. This representation reduces the number of operations required to perform a forward pass in the network, Figures 6.5(a) and 6.5(b).

Using this implementation results in a worse approximation to the FIM, discussed further in the next section, see Figure 6.8(a), though we did not observe meaningful differences in performance between the two implementations. The reduction in computational effort (as measured



((a)) Resource comparison of different implementations. The orange line is overlayed by the green line.



((b)) Reduction in average walltime resulting from the change of implementation. These values are the average of 1000 sampling steps and 100 energy computations of the model on 1 V100 GPU.

Figure 6.5: The blue line corresponds to the network outlined in Reference [204], the orange line to the method of splitting the single streams, Equations (6.40) and (6.41), and the green line the resource requirements of splitting the single stream and removing redundant pairwise streams. The resource requirements are measured as the number of required operations, n_{ops}, Equation (6.39). The walltime comparison of these methods is shown in (b). It is important to note that the computational time of the framework is dominated by the determinant calculation. These improvements will become negligible at much larger systems.

by the per iteration walltime as a proxy) was not as large as the drop in resource requirements, though was significant enough ($\sim 5-10\%$) to warrant the additional complication.

A complete set of the hyperparameters of the FermiNet* Ansatz used in these experiments are given in Table 6.1.

Spin for system X should be maximized (Hund's rule). Though it is not clear how these multi-electron orbitals should be interpreted, arranging the system in this way is better than other obvious methods (evenly splitting the number of electrons between spins), Figure 6.6.

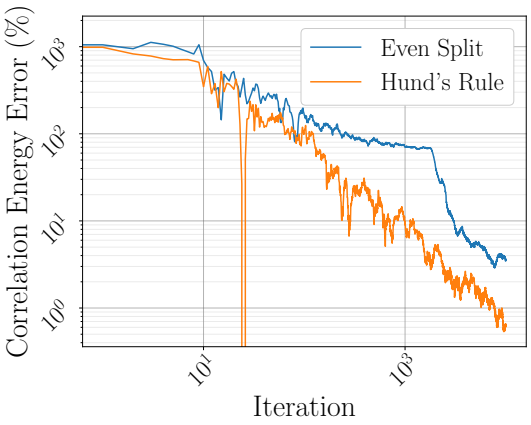


Figure 6.6: Demonstration of performance of the network dependent in the spin structure of the Ansatz.

6.3.2 Pretraining

The pretraining was performed very similarly to as described in Reference [204]. We outline the methods here for completeness.

The pretraining loss is

$$\mathscr{L}^{\text{pre}} = \int \Big[\sum_{\alpha\in\{\uparrow,\downarrow\}} \sum_{ijk} \Big(\phi_{ij}^{k\alpha}(X) - \phi_{i\alpha}^{\text{HF}}(\mathbf{r}^{\beta}) \Big)^2 \Big] p^{\text{pre}}(X) dX \tag{6.44}$$

where

$$p^{\text{pre}} = \frac{1}{2} \Big(\prod_{\alpha\in\{\uparrow,\downarrow\}} \prod_{i} \phi_{i\alpha}^{\text{HF}}(\mathbf{r}^{\beta}) + |\psi(X)|^2 \Big), \tag{6.45}$$

with $\phi_{i\alpha}^{\text{HF}}$ being Hartee-Fock orbitals obtained with STO-3g basis evaluated with PySCF python package.

This quantity is approximated by splitting the samples (at random) into two equal length sets and taking one Metropolis Hastings step with $p(X) = \prod_{\alpha\in\{\uparrow,\downarrow\}} \prod_i \phi_{i\alpha}^{\text{HF}}(\mathbf{r}^{\alpha i})$ for the first set and $p(X) = |\psi(X)|^2$ for the second set. The sets are joined and the process repeated. This implementation is slightly different to the original work, and seems to improve the pretraining: The Ansatz (after pretraining) has a lower energy.

As described in Reference [204], this distribution samples where the Hartree Fock (HF) orbitals are large, and also where the wave function is poorly initialized and is incorrectly large. This method of pretraining is a better solution to the problem of guiding the initial wave function. An Ansatz close to the HF orbitals seems to not get stuck in local minima and drastically improves the optimization behaviour. Other solutions include integrating the HF orbitals into the network

throughout the whole training, for example in Reference [108]. The optimization was performed using an Adam optimizer with default hyperparameters, notably learning rate 1×10^{-3}.

6.3.3 Kronecker Factored Approximate Curvature

We found divergent behaviour of the damping in the asymptotic region of the training, most likely due to the relatively large noise on the loss and small precision requirements of this particular optimization, though more rigorous investigation will likely yield interesting and useful insight.

As such, we use a reduced version of KFAC. We decay the learning rate, norm constraint and damping

$$x = \frac{x_0}{1 + t \times 10^{-4}} \tag{6.46}$$

where x stands in for the damping, learning rate and norm constraint and x_0 for the initial values. t is the iteration of the optimization. The norm constraint is particularly important at the start of training when the quadratic approximation to the local optimization space is large, the natural gradients are large and lead to unstable optimization. In this region the norm constraint plays a role and effectively clips the gradients. Later in the optimization the approximate natural gradients are small and not constrained. The damping / learning rate interplay is an essential ingredient to a functioning KFAC implementation. The algorithm has nice convergence properties, for this problem especially, but high performance is only possible with careful tuning of the damping and learning rate.

We bundle parameters in the envelope layers by using sparse matrix multiplications. We compute the entire Fisher block for all parameters of the same symbol (i.e. the $\boldsymbol{\pi}$ and $\boldsymbol{\Sigma}$ parameters). This is illustrated in Figure 6.8(b). Although we found this implementation was faster in practice, it performs more operations than maintaining a separation between these layers. It does not scale favorably for larger systems, though relative to the computational cost of the network may be negligible.

In Figures 6.8(a) and 6.8(b) we show the changes on the left Fisher factor resulting from using an implementation which splits data in the permutation equivariant function, versus an implementation which does not. For atomic systems, the right Fisher factor, $\bar{S}$, is the same in this new formulation. Therefore the changes in $\bar{A}$ are representative of the changes to the entire FIM.

The quality of approximation to the FIM is dependent on the data distribution and the model. In this problem and for a small model, we found that this approximation provided a better representation of the exact FIM than other methods [96] when compared on a small model over an initial optimization run of 1×10^3 iterations. It is not possible to perform the comparison to the exact FIM as the memory requirements to compute the exact FIM scale as $\mathcal{O}(n_{\mathrm{w}}^2)$ where n_{w} is the number of parameters.

6.3.4 Variational Monte Carlo with Kronecker Factored Approximate Curvature

The goal is to create a good enough approximation of the wave function and therefore the nodal structure of the wave function using VMC to be followed up with DMC given the FermiNet* Ansatz.

The VMC requires c_l forward passes (sampling), 1 backward pass (gradient) and 1 energy computations. To sample the wave function we use Metropolis Hastings. The step size is adaptively changed during training to move the acceptance toward the target sampling acceptance ratio of 0.5. c_l is the correlation length and here was set to 10. The model is pretrained for 1×10^3 iterations using the methods outlined in Section 6.3.2.

The gradients of the parameters are computed from Equation (5.43) and the centered energies $(E_L(X) - \mathbb{E}_X[E_L(X)])$ in Equation (5.43) have been clipped 5x from the median value and replaced with $\tilde{E}_L(X)$. Walkers X are sampled via Metropolis Hastings. A walker is a point in a space that moves around the space by taking random steps. These moves are accepted or rejected dependent on the difference in probability density of the starting and end points. A description of the algorithm is given in Algorithm 3.

Algorithm 3 Metropolis-Hastings algorithm used here for sampling. c_l is the correlation length. $\mathbf{r}$ and $\boldsymbol{xi}$ are $M \times n_e \times 3$ dimensional tensors. Steps update electron positions simultaneously and acceptance of steps are performed in parallel across all walkers. Line 7 updates walkers where the condition is true. σ is the step size that is adaptively determined to maintain an acceptance ratio of 0.5.

1: **for** c_l **do**
2: $\quad \boldsymbol{\xi} \sim N(0, \sigma)$
3: $\quad \mathbf{r}' \leftarrow \mathbf{r} + \boldsymbol{\xi}$
4: $\quad \alpha \sim U[0,1]$
5: $\quad P_{\text{move}} \leftarrow \frac{p(\mathbf{r}')}{p(\mathbf{r})}$
6: $\quad$ **if** $P_{\text{move}} > \alpha$ **then**
7: $\quad\quad \mathbf{r} \leftarrow \mathbf{r}'$
8: $\quad$ **end if**
9: **end for**

The approximate natural gradients for layer l, which are the updates to the Ansatz parameters, are computed via Equation (6.34). The approximation to the Fisher factors, $\bar{A}$ and $\bar{S}$, are 'warmed-up' by taking 100 steps of stochastic gradient descent with small learning rate ($\nu = 1 \times 10^{-5}$) whilst accumulating statistics. This ensures a smoothed initial approximation to the FIM and is consistent with other work in the literature [166].

For all systems, other than Beryllium, 1×10^5 iterations of VMC were run. The full VMC and KFAC algorithm used in this work is outlined in Algorithm 4 and a full set of hyperparameters and initialization values of variables are given in Table 6.2.

Name	symbol	initial value
iteration	k	0
number iterations	$k_{\max}^{\text{VMC}}$	1×10^5
learning rate	η	1×10^{-4}
norm constraint	c	1×10^{-4}
covariance state decay	κ	0.95
damping	λ	1×10^{-4}
damping factor	π	N/A
instantaneous activations	$\mathbf{a}$	N/A
instantaneous sensitivities	$\mathbf{s}$	N/A
energy gradients	$\Delta\mathscr{L}$	N/A
approx. natural gradients	$\tilde{\delta}$	N/A
variable decay	ν	1×10^{-4}
spatial locations	i	N/A
decaying average left Kronecker factor	$\bar{A}_l$	$\mathbf{0}$
decaying average right Kronecker factor	$\bar{S}_l$	$\mathbf{0}$
instantaneous left Kronecker factor	A_l	N/A
instantaneous right Kronecker factor	S_l	N/A
correlation length	c_l	10
sampling step size	σ	0.02^2
batch size	m	4096

Table 6.2: Table containing all variables used in the VMC algorithm using KFAC updates in this work.

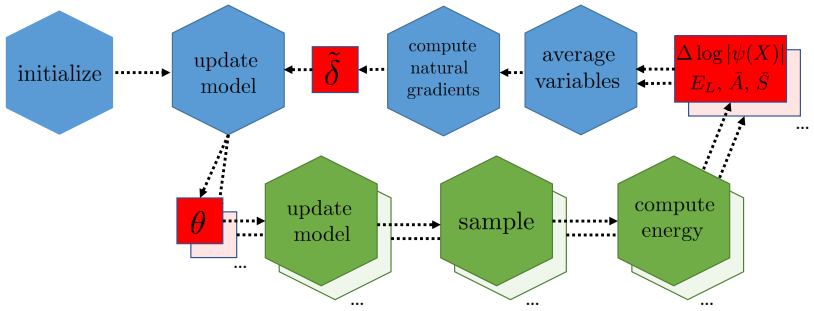
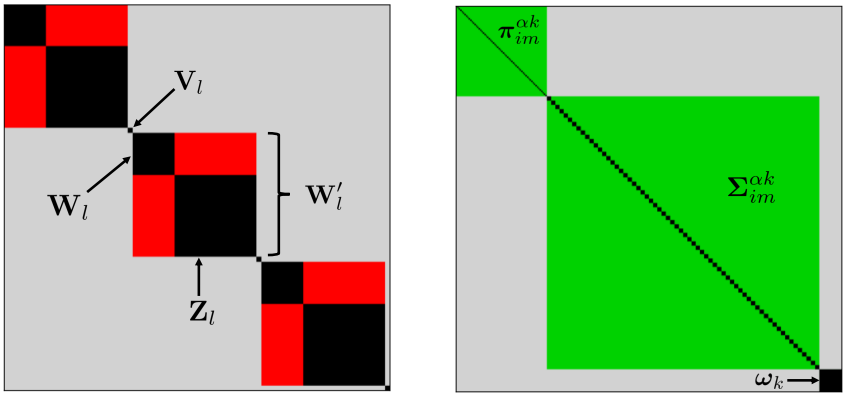


Figure 6.7: Flow diagram of distributed KFAC used in this work. The blue hexagons are the head worker, green hexagons workers. There are multiple workers, indicated by the staggered images and ellipses. The red squares are variables computed at one step and used at the next. θ are the parameters of the model, $\tilde{\delta}$ are the approximate natural gradients, $\Delta \log|\psi(X)|$ are the derivatives of the wave function wrt the electron position vectors and E_L are the local energies of the walkers X. These are tensors with M copies, where M is the batch size on a worker. $\bar{A}$ and $\bar{S}$ are the left and right Fisher factors computed during the forward and backward passes. These variables are averaged before being used to compute the approximate natural gradients. Note that the average $\bar{A}$ and $\bar{S}$ are *not* passed back to the workers. This results in a worse approximation to the Fisher, but a difference in performance was not noticed during tests.



((a)) Fisher factor A of only the layers containing $\mathbf{Z}_l$, $\mathbf{W}_l$ and $\mathbf{V}_l$ parameters. $\mathbf{W}'_l$ represents the variables for computing the Fisher factor from a layer where the parameters $\mathbf{W}_l$ and $\mathbf{Z}_l$ are not split into two layers.

((b)) Fisher factors A of the layers corresponding to the $\boldsymbol{\pi}^{\alpha k}_{im}$, $\boldsymbol{\Sigma}^{\alpha k}_{im}$ and $\boldsymbol{\omega}_k$ parameters.

Figure 6.8: Two cartoons of the differences between the approximations to the 'left Fisher factor' A, Equation (6.34), in this work and in Reference [204]. Regions in gray are not computed in either case. Black are computed in both cases. Red are regions lost in this work and green are regions gained. The variable labels (e.g. $\boldsymbol{\pi}^{\alpha k}_{im}$) are the parameters corresponding to the Fisher factor.

6.3.4.1 KFAC Hyperparameter Optimization

KFAC is a high performance approximate second order method optimization method. The authors of the original work, Martens and Grosse [96], likened 2nd order methods to a race car, "As an analogy one can think of such powerful 2nd-order optimizers as extremely fast racing cars that need more sophisticated control systems than standard cars to prevent them from flying off the road. Arguably one of the reasons why high-powered 2nd-order optimization methods have historically tended to under-perform in machine learning applications, and in neural network training in particular, is that their designers did not understand or take seriously the issue of quadratic model approximation quality, and did not employ the more sophisticated and effective damping techniques that are available to deal with this issue.". We experienced much difficulty achieving reliable performance and understanding the heuristics for hyperparameter setting. What follows is the hyperparameter tuning of KFAC for this problem and network. All experiments, unless otherwise stated, were performed on a 'half-model': 2 hidden layers, 128 hidden units in the single electron streams and 16 hidden units in the pairwise electron streams. Table 6.3 details the general hyperparameters used in all of these experiments, unless otherwise stated in the caption of the relevant figure.

Each experiment is a single run taken over a small number of iterations. Under ideal conditions, we would take many runs and show both the initial and asymptotic behaviours.

Name	symbol	value
# Single Stream hidden units	n_{s}	128
# Pairwise Stream hidden units	n_{p}	16
# Split Stream hidden units	n_{ss}	128
# determinants	n_k	8
# layers	n_{l}	2
# learning rate	η	1×10^{-4}

Table 6.3: Model hyperparameters used for the hyperparameter optimization.

However, the realities of time and space require that we limit these experiments. Therefore, they should be taken as rough indicators, which may or may not show true behaviours.

There will be differences in behaviour between a 'half-model' and the full model, Table 6.2, but we were restricted in the available compute resources. For example, the quality of approximation to the local curvature will be different. Understanding exactly how they are different and precise rules for the implementation of KFAC is no simple task, with potentially no clear answer. As such, we use the following results to guide our implementation of the full model.

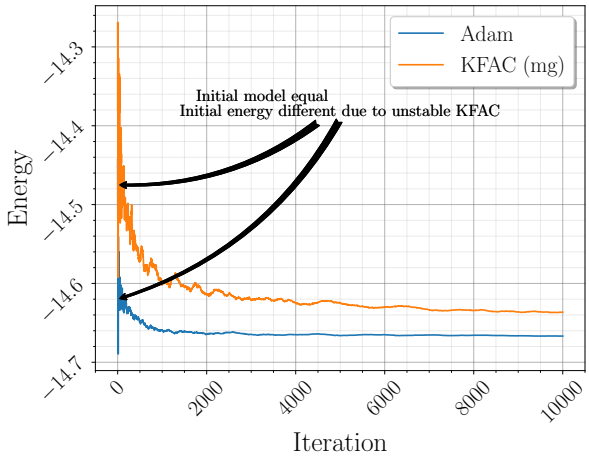
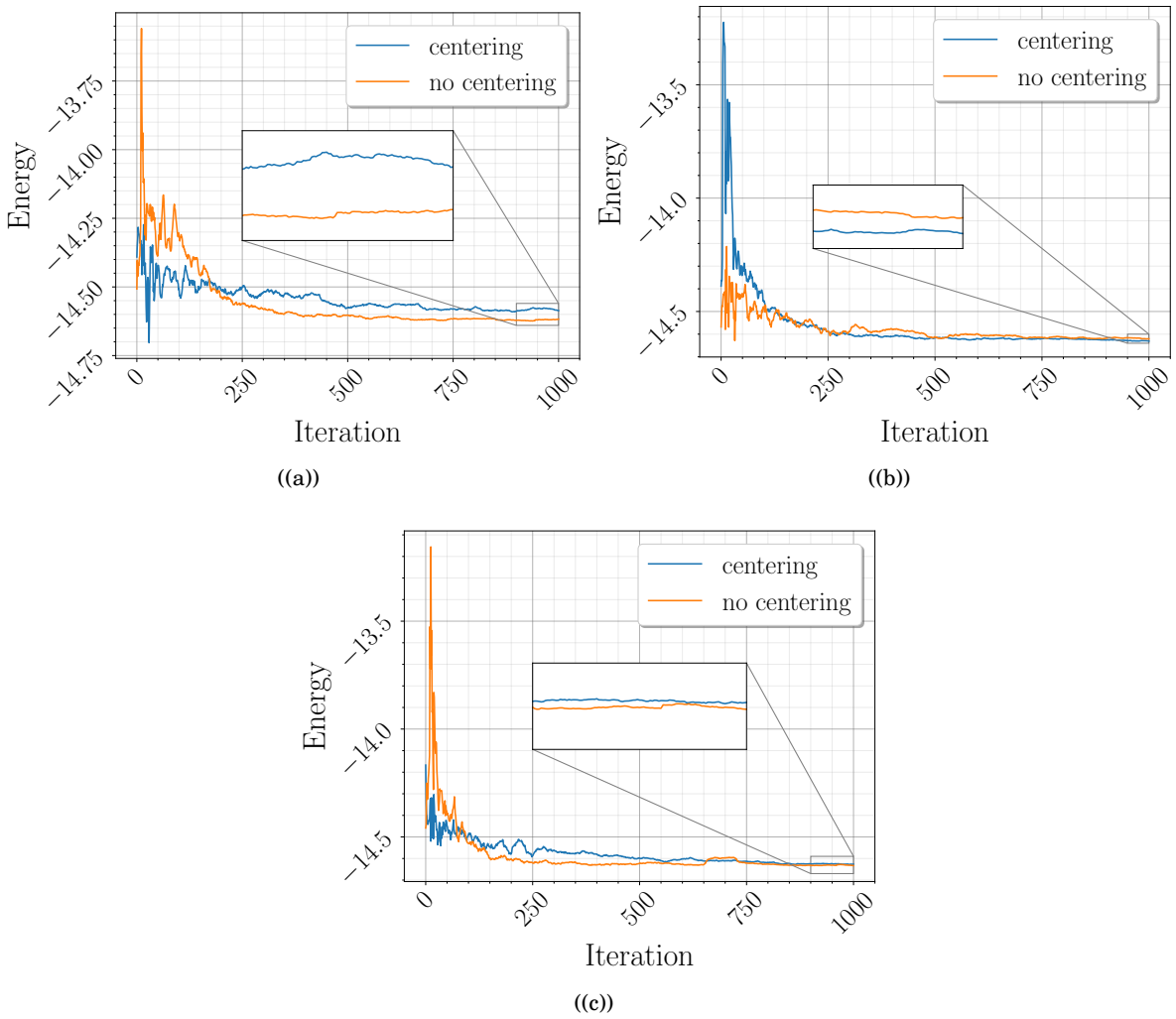


Figure 6.9: Initial comparison of KFAC and ADAM optimizers described in [204]. These experiments used the full model. The (mg) label indicates the convolutional approximation described in Reference [96].

The first step was attempting to replicate previous work in the field. Figure 6.9 shows a comparison between KFAC and ADAM optimizer, using the exact implementation and hyperparameters detailed in Reference [204]: These results do not match. In previous work, using these hyperparameters, KFAC outperforms ADAM, but as we can see in Figure 6.9 ADAM is clearly the better choice. There are two possible reasons for this discrepancy. Though our implementation was carefully validated against other implementations [164], it is possible there are differences between our model or KFAC implementation and other work. Second, our implementation uses PyTorch and Cuda 10.1, whereas the previous work used Tensorflow 1.4 and CUDA 9.0. As part of

the validation we implemented a different version in Tensorflow 2.0 and CUDA 10.1. We observed numerical differences between Tensorflow 2.0 with CUDA 10.1 and Pytorch with CUDA 10.1, it is possible that differences in the frameworks lead to different optimization behaviours. Though the exact reason for this was never discovered, through careful hyperparameter optimization we managed to achieve performance equal to or beyond that in previous work.



((a)) ((b))

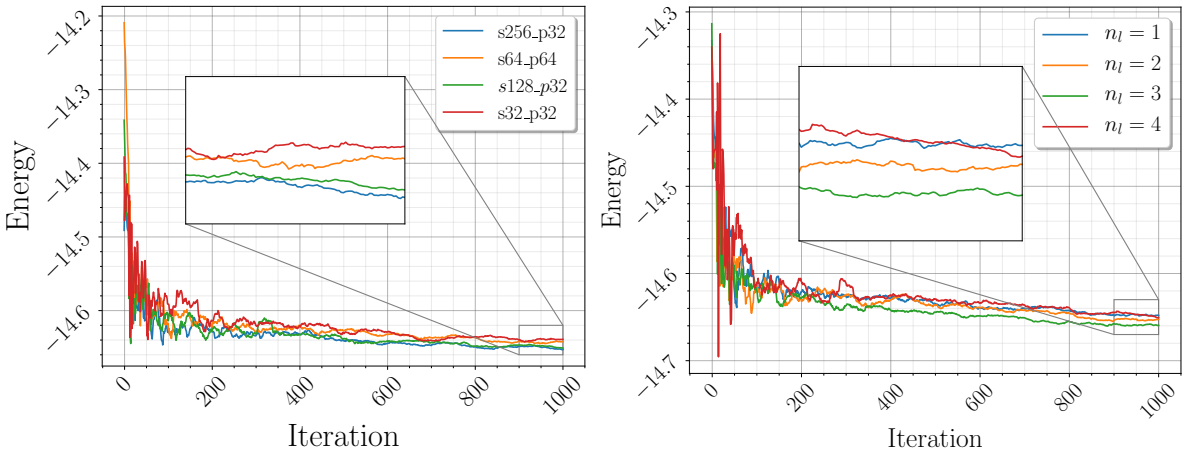
((c))

Figure 6.10: Three plots comparing the effect of centering on the convolutional approximations (a) 'mg' Reference [96], (b) '*ba*' Reference [22] and (c) '*bg*', a combination of these methods. The labels '*mg*', '*ba*' and '*bg*' are convenient shorthand.

In Figure 6.10 we compare the differences in convolutional approximation and the affect of centering, described in Section 6.2.3. In subfigures a, b and c we show three different convolutional approximations: '*mg*', '*ba*' and '*bg*', respectively. The *mg* label indicates methods outlined in Reference [96], *ba* in Reference [22] and *bg* is a mix of these methods. Roughly speaking, the mg method approximates the local curvature around all parameters by assuming uncorrelated

activations and sensitivities and uncorrelated data streams, i.e. $\mathbb{E}[\mathbf{a}_0\mathbf{a}_1] = \mathbb{E}[\mathbf{a}_0]\mathbb{E}[\mathbf{a}_1]$. The *ba* method simplifies this and approximates the activations and sensitivities used in Equation (6.18) by taking the average over all data streams. Finally, the *bg* method intuitively combines these methods by assuming the 'up' and 'down' data streams, that is the data coming from electrons of the same spin, are more correlated than those from different spins. In parts of the network where parameters are involved in both streams, the mean contribution from each is taken and the resulting averages are passed into the mg method. Though this method appears to at least equal the performance of the *ba* method, it is more computationally demanding and requires more motivation, showing or proving the correlations in the data streams.

In these experiments we also compare the affect of centering. Explicitly the centering is required, for example to match the derivation of Stochastic Reconfiguration given an unnormalized wave function, but we find that the performance without centering is either equal to or greater than with centering. There is no concrete description of why this should be the case. We note this here and pose the following questions for future work: Will the uncentered versions of KFAC generalize to larger and more complicated systems? Why is the performance of the uncentered version at least equal to the centered version? Given the performance, computational advantages and relative simplicity of the implementation, the *ba* method was chosen for this work.



((a)) ((b))

Figure 6.11: Two plots comparing the effect of changing the size of the network. In (a) the numbers following s and p indicate the number of units in the hidden layers of single and pairwise streams, respectively. In (b) n_l is the number of hidden layers.

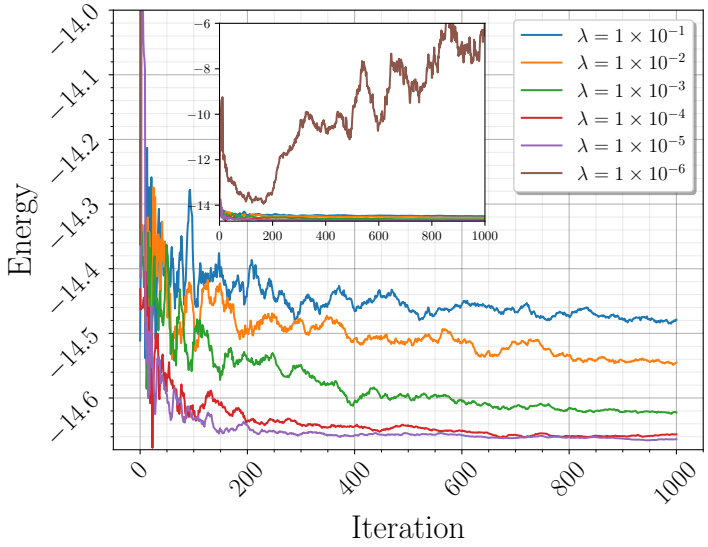
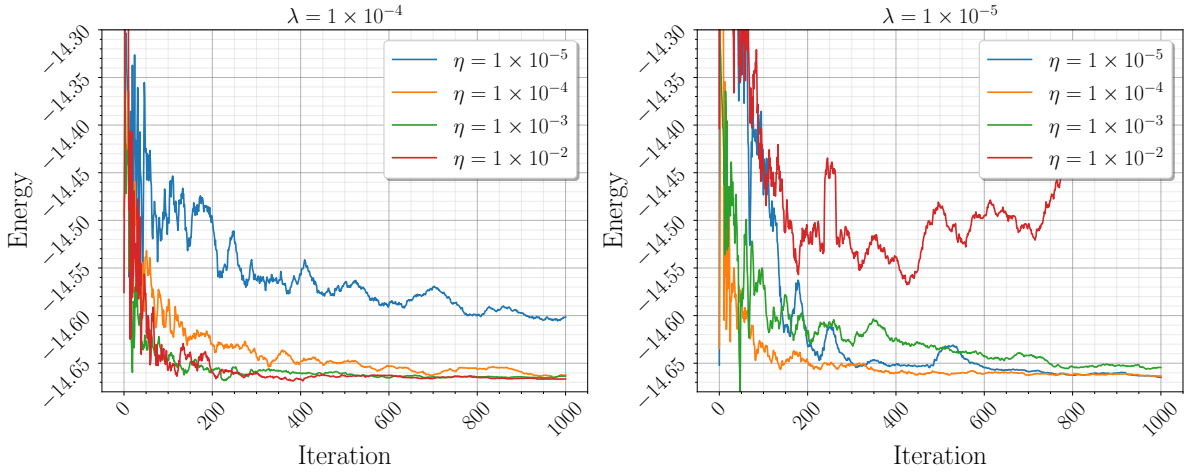


Figure 6.12: Optimization of the damping, here denoted by λ.

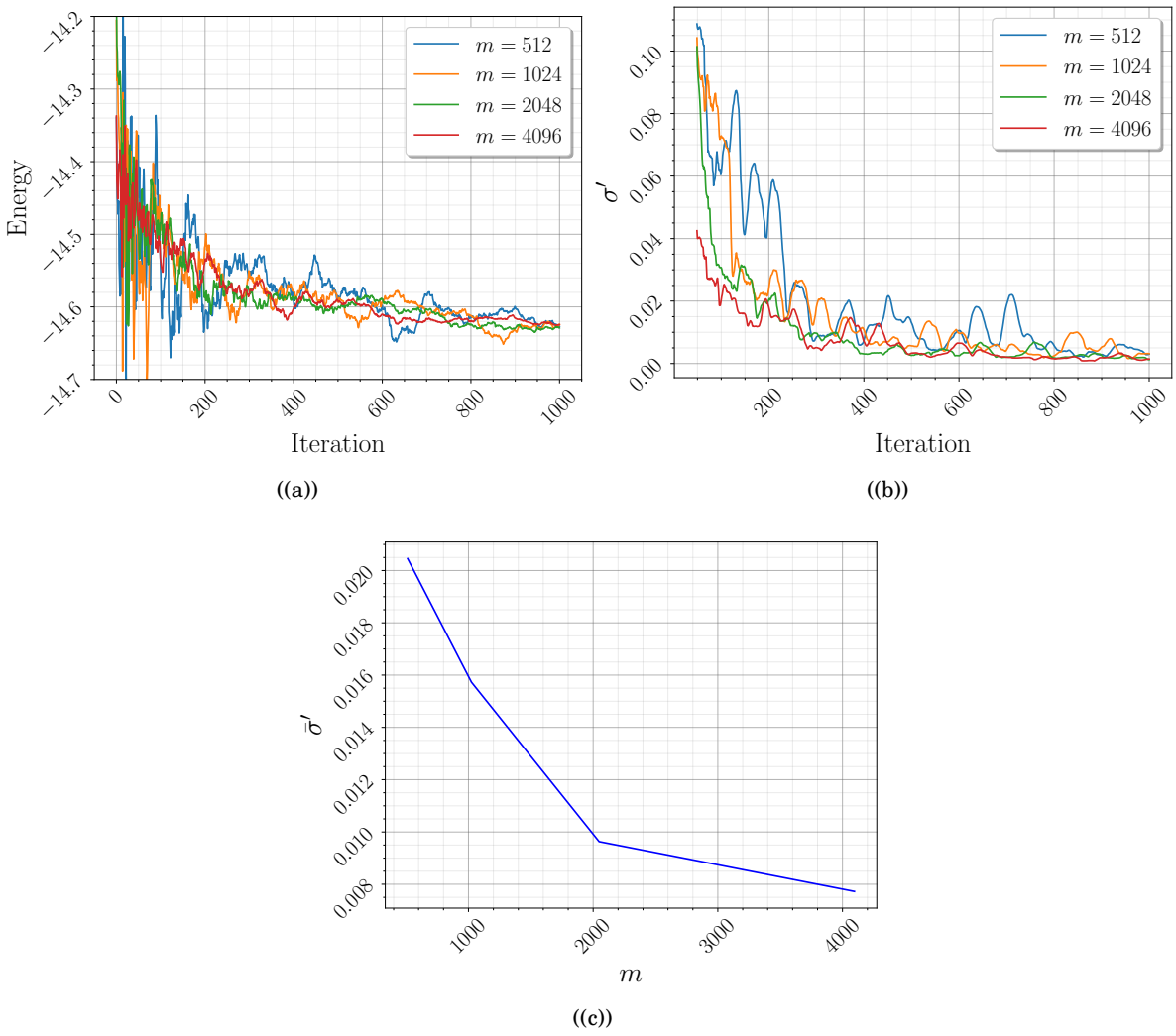
Figure 6.11 shows the network structure optimization. These results indicate larger hidden layers are more effective, which is in-line with general neural network intuition, and relationship of the depth with the performance is nonlinear. Figure 6.12 explores the damping: Large damping is stable and results in slow optimization whereas small damping can result in fast optimisation but divergent behaviours.



((a)) ((b))

Figure 6.13: Optimization of the leanring rate, here denoted by η, over damping (a) 1×10^{-4} and (b) 1×10^{-5}, here denoted by λ

Figure 6.13 demonstrates the large dependence of the algorithm on the learning rate, following a similar pattern to the learning rate: Large learning rates were fast but unstable.



((a)) ((b)) ((c))

Figure 6.14: Three plots showing the effect of different batch sizes on the optimization. (a) shows the raw data, (b) plots σ', the standard deviation of the energy within a moving window of 100 iterations, and (c) the mean σ' over the entire optimization as a function of the batch size. Larger batch size results in more stable updates.

The performance of KFAC is strongly dependent on the batch size of the data. A small batch size causes in a bad approximation of the Fisher, which can result in unstable updates. A larger batch size improves both the approximation of the gradients and the Fisher. Though Figure 6.14 (a) indicates there is little difference in the final energies of tested batch sizes, Figure 6.14 (b) shows σ', the standard deviation of a moving window of the energies. This measure reduces as the optimization progresses, because the size of the updates decreases, so more precise measures are needed to assess the behaviour of the algorithm with different batch sizes in the asymptotic region.

Name	symbol	initial value
number iterations	$k_{\text{max}}^{\text{DMC}}$	5e4
Forces	$\mathbf{F}$	N/A
Green's function	$G(\mathbf{r},\mathbf{r}')$	1.
trial energy	E_T	N/A
acceptance (rejection) probability	p (q)	N/A
weights	$\boldsymbol{\omega}$	$\mathbf{1}$
batch size	M	4096

Table 6.4: Variables used in the Diffusion Monte Carlo method here.

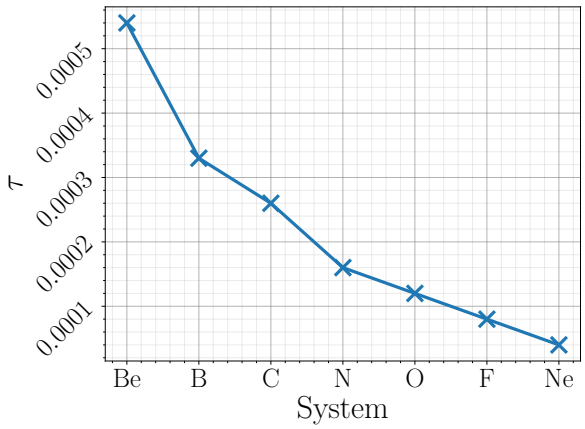


Figure 6.15: All τ generated for each system studied in this work.

6.3.5 Diffusion Monte Carlo

We use a reduced version of the DMC algorithm outlined in Reference [257] and the full algorithm used in this work is outlined in Algorithm 6.3.5. We make simplifying changes by removing transformations to spherical coordinate systems and set $\tau_{\text{eff}} = \tau$. Also, we move all walkers and electrons simultaneously with no change to how the algorithm behaves. We set τ by testing a non-uniform range of values for τ, computing the average acceptance over 100 iterations, and solving the equation of a line between two points to find the closest value that gives an acceptance ratio (Acceptance) of 0.999, shown in Figure 6.16. We found this worked well in practice and all τ acceptances were $\sim 99.9\%$. Figure 6.15 shows all τ generated for each system studied in this work. A complete list of the hyperparameters needed given in Table 6.4.

The DMC implementation is parallelized over all electrons and walkers. The model weights and walker configurations are converted to 64-bit floats for the DMC. DMC was run for a variable number of iterations, where the minimum was 5×10^4, and this is discussed further in Section 6.5.

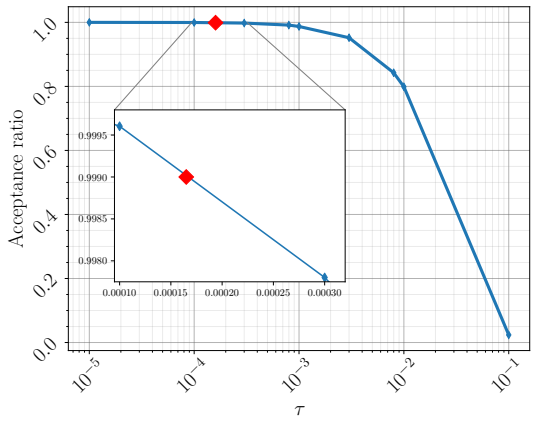


Figure 6.16: The Acceptance ratio of Nitrogen as a function of the time-step τ. The red diamond marker indicates the estimated value of τ at an acceptance ratio of 0.999 (99.9%).

6.3.6 Code and Hardware

Each VMC experiment used 2 V100 GPUs, the DMC runs were distributed over a variable number of CPUs due to constraints on the availability of GPU time.

The code was written in PyTorch 1.5 using CUDA 10.1. The implementation was parallelized such that each GPU held 2 models, or each CPU 1 model. Initial attempts at producing this model were written in Tensorflow 2 with CUDA 10.1 but we experienced significant and sometimes not (easily) diagnosable issues. A simplified version of this Ansatz will be released to DeepQMC [259].

Computations of the Hartree Fock orbitals were performed using PySCF [247] and the model was distributed using Ray [2].

6.4 Results and Discussion

Diffusion Monte Carlo

We demonstrate functionality of the introduced computational techniques on the atomic systems from second period (Be-Ne), and the cation C^+. For each system, first we optimize parameters of the FermiNet*, in order to generate a good trial wave function, that is subsequently improved through DMC method. As it was demonstrated in [204] FermiNet is capable of outperforming other VMC methods. Our VMC results do not exceed the existing state-of-the-art, the DMC either exceed or match other best results (see Table 6.4).

Algorithm 4 KFAC for FermiNet*. **CholeskyInverse** indicates the Cholesky Inversion, **FactoredTikhonov** is the technique for computing π given by Equation (6.23), and **Clip** indicates clipping a batch of values to within 5x of the median value. A high level overview of the distribution of this algorithm across multiple workers is shown in Figure 6.7

for $k_{\text{max}}^{\text{VMC}}$ **do**

  Update walker coordinates X ▷ Metropolis Hastings
  $\tilde{E}_L = \textbf{Clip}(E_L(X) - \mathbb{E}_X[E_L(X)])$ ▷ energy
  $\Delta\mathscr{L} = \mathbb{E}_X\left[\tilde{E}_L \nabla \log|\psi(X)|\right]$ ▷ backward pass
  Compute $\mathbf{a}$ ▷ forward pass
  Compute $\mathbf{s}$ ▷ backward pass

  for all layers l **do**

    $\bar{\mathbf{a}}_l \leftarrow \mathbb{E}_i[\mathbf{a}_{li}]$
    $A_l \leftarrow \mathbb{E}_X[\bar{\mathbf{a}}_l^T \bar{\mathbf{a}}_l]$
    $\bar{A}_l \leftarrow \kappa \bar{A}_l + (1-\kappa) A_l$

    $\bar{\mathbf{s}}_l \leftarrow \mathbb{E}_i[\mathbf{s}_{li}]$
    $S_l \leftarrow \mathbb{E}_X[\bar{\mathbf{s}}_l^T \bar{\mathbf{s}}_l]$
    $\bar{S}_l \leftarrow \kappa \bar{S}_l + (1-\kappa) S_l$

    $\pi_l \leftarrow \textbf{FactoredTikhonov}(\bar{A}_l, \bar{S}_l)$
    $\lambda_A \leftarrow (\frac{\lambda}{\pi \times |T|^2})^{1/2}$
    $\lambda_S \leftarrow (\frac{\lambda \times \pi}{|T|^2})^{1/2}$

    $\bar{A}_l^{-1} \leftarrow \textbf{CholeskyInverse}(\bar{A}_l + I\lambda_A)$
    $\bar{S}_l^{-1} \leftarrow \textbf{CholeskyInverse}(\bar{S}_l + I\lambda_S)$

    $\tilde{\delta}_l \leftarrow \bar{A}_l^{-1} \frac{\Delta_l \mathscr{L}}{|T|^2} \bar{S}_l^{-1}$

  end for

  $\epsilon \leftarrow \min\left(1, \sqrt{\frac{c}{\sum_l \tilde{\delta}_l \Delta_l \mathscr{L}}}\right)$

  for all layers l **do**

    $\theta_l \leftarrow \theta_l - \epsilon \times \eta \times \tilde{\delta}_l$ ▷ Update the model

  end for

  $\eta \leftarrow \frac{\eta_0/\nu}{1+k}$
  $\lambda \leftarrow \frac{\lambda_0/\nu}{1+k}$
  $c \leftarrow \frac{c_0/\nu}{1+k}$

end for

Algorithm 5 Diffusion Monte Carlo. $\mathbf{r}$, $\boldsymbol{\xi}$ and $\mathbf{F}$ are $M \times n_e \times 3$ dimensional tensors. Steps update electron positions simultaneously and acceptance of steps are performed in parallel across all walkers. Line 7 updates walkers where the condition is true.

1: Compute $\psi(X)$ ▷ forward pass
2: $\mathbf{F} \leftarrow \frac{d \log|\psi(X)|}{d\mathbf{r}}$ ▷ backward pass

3: **for** $k_{\text{max}}^{\text{DMC}}$ **do**

4: $G(\mathbf{r}, \mathbf{r}') = G(\mathbf{r}', \mathbf{r}) = 1$

5: $\boldsymbol{\xi} \sim N(0, \tau)$
6: $\mathbf{r}' \leftarrow \mathbf{r} + \tau\mathbf{F} + \boldsymbol{\xi}$

7: $X' \leftarrow \{\mathbf{R}, \mathbf{r}'\}$
8: Compute $\psi(X')$ ▷ forward pass
9: $\mathbf{F}' \leftarrow \frac{d \log|\psi(X')|}{d\mathbf{r}'}$ ▷ backward pass

10: $G(\mathbf{r}, \mathbf{r}') \leftarrow \prod_i^{n_e} \exp\left((\mathbf{r}_i - \mathbf{r}'_i - \tau\mathbf{F}'_i)^2/2\tau\right)$
11: $G(\mathbf{r}', \mathbf{r}) \leftarrow \prod_i^{n_e} \exp\left((\mathbf{r}'_i - \mathbf{r}_i - \tau\mathbf{F}_i)^2/2\tau\right)$

12: $p \leftarrow \min\left(1, \frac{|\psi(X')|^2 G(\mathbf{r}, \mathbf{r}')}{|\psi(X)|^2 G(\mathbf{r}', \mathbf{r})}\right)$
13: $q \leftarrow 1 - p$
14: Set $p = 0$ where $\text{sign}(\psi(X)) \neq \text{sign}(\psi(X'))$ ▷ Fixed node approximation

15: $s \leftarrow E_T - E_L(X)$
16: $s' \leftarrow E_T - E_L(X')$
17: $\omega \leftarrow \omega \times \exp\left[\tau\left[\frac{p}{2}(s' + s) + qs\right]\right]$
18: $\alpha \sim U[0, 1]$
19: **if** $p > \alpha$ **then**
20: $\psi(X) \leftarrow \psi(X')$, $\mathbf{r} \leftarrow \mathbf{r}'$, $\mathbf{F} \leftarrow \mathbf{F}'$, etc ▷ Update variables for next iteration
21: **end if**
22: **end for**

Atom	FermiNet* + DMC (This work)	FermiNet* (This work)	FermiNet [204]	VMC [235]	DMC [235]	HF	Exact [58]
Be	-14.66734(2)	-14.66726(1)	-14.66733(3)	-14.66719(1)	-14.667306(7)	-14.35188	-14.66736
B	-24.65384(9)	-24.65290(2)	-24.65370(3)	-24.65337(4)	-24.65379(3)	-24.14899	-24.65391
C^+	-37.43086(6)	-37.43061(1)	-37.4307(1)†	-37.43034(6)	-37.43073(4)	-36.87037	-37.43088
C	-37.84472(7)	-37.84452(1)	-37.84471(5)	-37.84377(7)	-37.84446(6)	-37.08959	-37.8450
N	-54.5891(5)	-54.58755(6)	-54.58882(6)	-54.5873(1)	-54.58867(8)	-53.5545	-54.5892
O	-75.0667(3)	-75.0599(2)	-75.06655(7)	-75.0632(2)	-75.0654(1)	-73.6618	-75.0673
F	-99.7332(5)	-99.7277(1)	-99.7329(1)	-99.7287(2)	-99.7318(1)	-97.9865	-99.7339
Ne	-128.9370(3)	-128.9351(1)	-128.9366(1)	-128.9347(2)	-128.9366(1)	-126.6045	-128.9376

Table 6.5: Comparison of VMC ad DMC results with existing works. FermiNet* energies are computed from 1×10^4 batches of size 8096 given a model after 1×10^5 of training. The number in the brackets indicates the error on the calculation at the same precision as the value reported. For example, -14.66734(2) indicates a value of -14.66734±0.00002. All errors reported in this work are the standard error on the mean.

† result not directly reported. Computed from ionisation energy and carbon energy result and the errors propagated via addition.

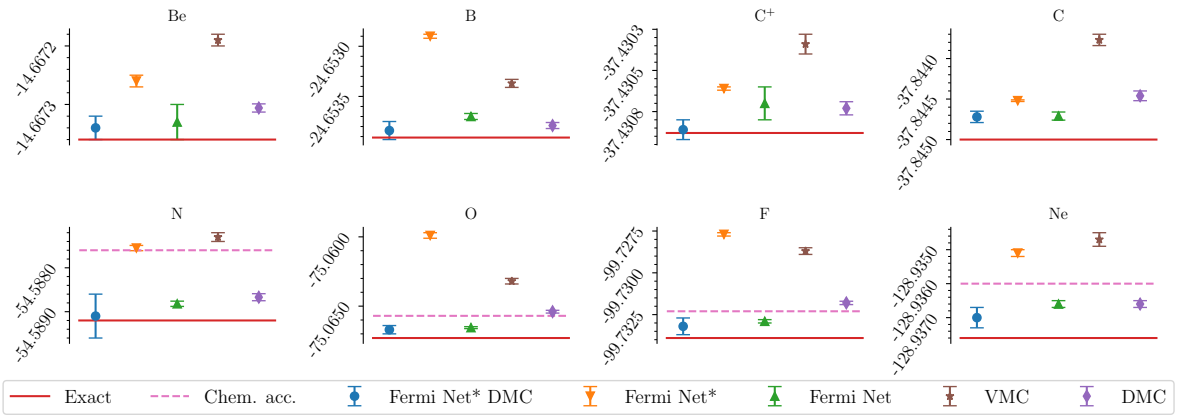


Figure 6.17: Graphical representation of data from Table 6.4, see caption for details. A chemical accuracy line (Chem. Acc. - pink dashed line) is plotted where it falls within the range of the plotted data for a system.

Table 6.4 summarizes the results. All references to FermiNet* Ansatz indicate the methods outlined in this work. The first column of this Table contains the final energies and errors of the wave function after both VMC and DMC with the FermiNet* Ansatz and the second column are the corresponding energies only after the VMC. The third through seventh columns (FermiNet, VMC, DMC, HF, and Exact) contain benchmark results from the literature indicated. The HF column are the Hartree-Fock energies obtained using the STO-3g basis, the basis used in this work for the pretraining orbitals.

The FermiNet* VMC energies in column 2 are not directly comparable to the FermiNet energies. We focused computational resources on the DMC and restricted all but one system (Be) to 1×10^5 iterations, whereas the original FermiNet work used double this number. We extended one Be to 2×10^5 iterations and found improvements to the energy, Figure 6.4, computed as -14.66730(1). This is at the top end of the error in the original FermiNet result, indicating that the performance may be matched with this implementation, if slightly worse. Highlighting this result clearly does not guarantee equivalent performance on larger systems and more extensive trials on larger systems are required. The other FermiNet* results are consistently poorer than FermiNet, but better than the referenced benchmark VMC results.

All energies in column 1 are to within less than 0.25% of the respective correlation energies, at best (Be) within 0.03%. Be, B, C^+, and N are all accurate to the exact ground state energy within error bars. Systems Be, B, C^+ and Ne are the most accurate energies. Systems Be, C^+, C and Ne were run with around 2×10^5 iterations, which are explored in more depth in Figure 6.4, and B, C, N, O with around 5×10^4 iterations of DMC.

The error bars are the standard error $\sigma_{\text{SEM}} = \sigma/\sqrt{m}$ where σ is the standard deviation of the energies of the batches and m is the number of batches evaluated. They are noticeably larger in column 1 of Table 6.4 for the systems mentioned where less iterations of DMC are run. DMC

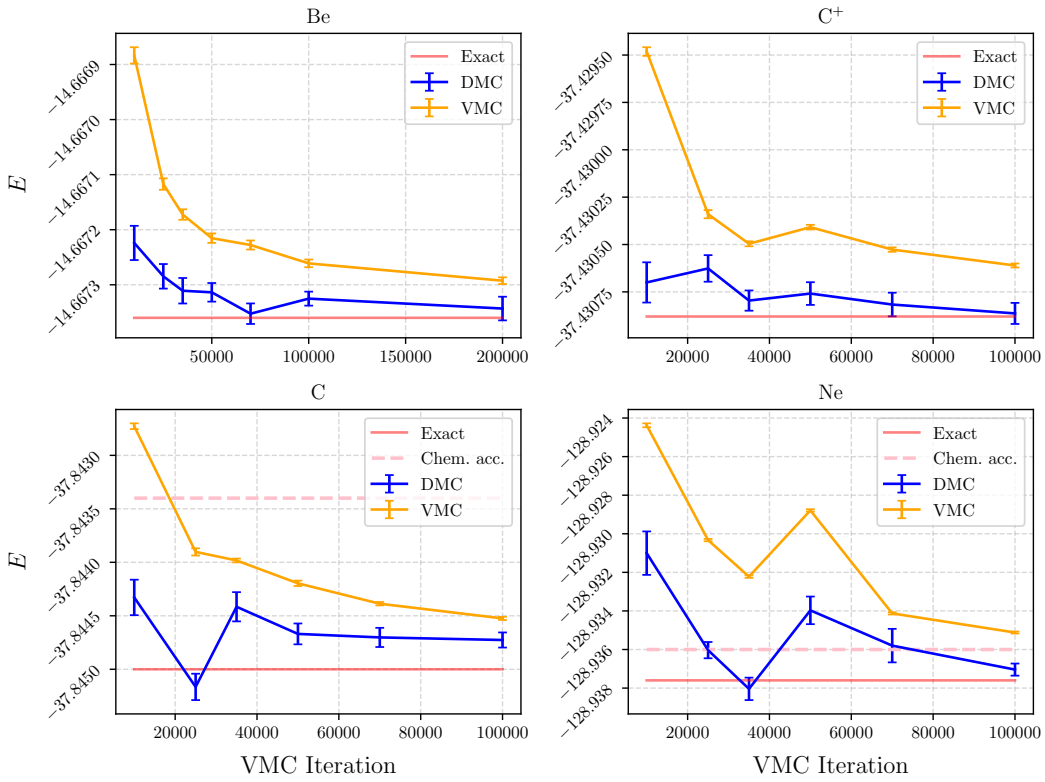


Figure 6.18: DMC applied to Ansatz at different stages of VMC optimisation. The red dashed line is the exact energy of the ground state, given in Table 6.4. The orange line is the energy of the wave function at a particular VMC iteration. Note that Be extends to 2×10^5. It is demonstrated here to show that there is still capacity for the wave function to improve, but we were restricted by computational time for the other systems. The blue line is the resulting energy from the application of DMC to that iteration of the wave function. The DMC converged generally after around 5×10^3 iterations, and was run for between $5 \times 10^4 - 2 \times 10^5$ iterations after convergence. The error bars are the standard error $\sigma_{\text{SEM}} = \sigma/\sqrt{m}$ where σ is the standard deviation of the energies of the batches evaluated and m is the number of batches. A chemical accuracy line is added to the plots where it is within the range of the plotted data.

consistently improved the wave function, that is the energy average was lower, but the error bars are significantly larger in some cases, due to the autocorrelation of the data trace.

Figure 6.4 shows DMC applied to different trial wave functions. Each DMC point is evolved from the corresponding VMC iteration. The VMC lines in general show a clear trend for improving the energy of the wave function as the iterations increase, and this is mirrored in the energy computed from the DMC. There are several apparent anomalies in the data. In the Neon VMC line there is a clear decrease in the quality of the model (increase in the energy) during the training. We believe this is a result of divergence in training and may indicate that our implementation of KFAC becomes more unstable as the molecules become larger. We cannot extrapolate significant conclusions without more analysis of the optimization. In both the Neon and Carbon DMC lines there is a large kink in the energy achieved by DMC. Further investigation did not reveal divergent behaviour in the DMC, that is large changes in energy after convergence, and repeated runs showed more stable behaviour in line with what would be expected. It is possible on these particular runs the walkers became trapped near the nodes and accumulated anomalous low energy statistics. Again, further analysis and development of this DMC is required to understand if this is a feature of the precise Ansatz or issues with the DMC implementation.

6.4.1 GPU, CPU and Computational Time

In this work we have used a combination of GPU and CPU compute resources. Typically, research groups may not have access to state-of-the-art compute architectures and may need to exploit alternate compute resources. The GPU methods were significantly faster than the CPU experiments, especially in the cases of larger systems, for example GPU vs. CPU per iteration times for the DMC algorithm on the system Neon were 4s and 24s, respectively, we found that in these small systems a CPU implementation distributed over 3 nodes was enough for reasonable experiment time (< 1 week). However, the scaling makes the CPU implementation impractical for larger systems. CPU implementations are not standard for the machine learning community, but are for the quantum chemistry community. We highlight here that these methods can be used on CPU architectures, though may quickly become impractical for larger systems.

One iteration of DMC is less computationally demanding than VMC, requiring 1 forward pass, 1 energy computation and 1 backward pass, plus some negligible functions. However, we port the FermiNet weights to 64-bit precision for the DMC phase, finding significantly better performance. This is increases the walltime of 1 iteration of DMC to 1-1.5x 1 iteration of VMC.

6.5 Conclusions

In this work we have changed the structure of a neural network Ansatz, the FermiNet, by removing redundant elements (the diagonal elements of the pairwise streams) and splitting the data in the permutation equivariant function such that it is not reused unnecessarily. These

changes increase the performance (as measured by the walltime) of the network. Additionally, we have improved approximations to the ground state, found with Variational Monte Carlo and a FermiNet* Ansatz, with Diffusion Monte Carlo, matching or exceeding state-of-the-art in all systems.

With respect to the first contribution, although this model is small compared with other state-of-the-art neural networks [45], it contains the expensive determinant computation, which is the dominant term in the complexity of the network scaling as $\mathcal{O}(n_e^3 k)$, where n_e is the number of electrons and k is the number of determinants. Variational Monte Carlo requires computation of the Laplacian, which uses n_e backward passes of the first order derivatives. In total, the estimated complexity of the network is $\mathcal{O}(n_e^4 k)$. Although the changes made here improve the performance, this improvement in speed may be negligible at larger system sizes.

6.5.1 Related work

Other more recent work [245] additionally found significant efficiency gains in altering the neural network with no noticeable reduction in accuracy: replacing the anisotropic decay parameters with isotropic decay parameters in the envelopes; and removing more redundant parameters in the determinant sum. Further, other smaller networks employing a more traditional Jastorw/backflow Ansatz [108] and integrated Hartree-Fock orbitals were significantly less computationally demanding at the cost of notably worse performance. There are clearly still gains to be made in the efficiency of these methods, and a better understanding on the relationship between the problem and the size of the network required will be important pieces of understanding for tuning these methods.

6.5.2 Future Work

In this work we alternate between 32-bit and 64-bit computational precision for the Variational Monte Carlo and Diffusion Monte Carlo methods, respectively. It may be possible to exploit this further. One example is mixed precision algorithms and networks [175]. A suitable and easy first step might be mixed precision schedules. Using low precision weights earlier in training and switching to high precision in the asymptotic region of the optimization might encourage even better results, for example improving the approximation to the Fisher Information Matrix in the region where there are small variations in the optimization landscape. In early tests, reducing the algorithm and model to 16-bit precision proved unstable by generating singular Fisher blocks (that could not be inverted).

There is a lot of room to improve the optimization. KFAC is a powerful algorithm which requires careful tuning. The methods used in this work are relatively simple, notably the schedule of the learning rate and damping, and not representative of the suggested mode of operation. Stable adaptive techniques will be essential to improving the optimization in this domain. Additionally, the efficiency of KFAC can be improved further by using intuitions about the FIM.

For example, the FIM will change less in the asymptotic region of the optimization, schedules which exploit this fact and update the Fisher blocks and their inverses less should be used, as outlined in the literature [166].

Finally, there are existing approximations from VMC literature which can be easily integrated into these frameworks, such as pseudopotentials (for scaling to larger systems) [208], and alternate representations of the wave function [24, 56]. Additionally, these new and highly precise techniques can be applied to other systems including solids and other interesting Hamiltonians.

6.5.3 Final comment

We conclude that though these results are comparatively good in isolation, the prospects for improvements in these methods and hardware paint a strong future for the application of neural networks in modelling wave functions in the continuum.

CHAPTER 7

CONCLUDING REMARKS

It's not a silly question if you can't answer it.
— Jostein Gardner, *Sophie's World*

The work in this Thesis explored methods from quantum machine learning. Chapter 3 established a hybrid quantum-classical framework for integrating near-term quantum devices. Chapter 4 introduced near-term heuristic quantum algorithms (VQE and QAOA), evaluated the performance of different optimizers and metalearning is shown to have behaviours that may motivate its usage, especially on noisy near-term hardware. Both these pieces of work introduce and expand the ways that quantum devices can be used in conjunction with classical machine learning frameworks. Finally, the theory underlying neural network Ansätze in quantum Monte Carlo was established in Chapter 5 and applied in Chapter 6, finding state-of-the-art performance on small atomic systems.

Contributions

- Developed of a hybrid quantum-classical neural network framework: the first quantum-assisted generative adversarial network and first (known) application to a complex color dataset (Chapter 3)
- Provided evidence to support neural networks as optimisers for heuristics in near-term hardware (Chapter 4)
- Improved state-of-the-art neural network methods for quantum Monte Carlo on small systems by altering the network design and extending with diffusion Monte Carlo (Chapters 5 and 6). This is the first, to our knowledge, demonstration of diffusion Monte Carlo with neural network methods.

7.1 Hybrid quantum-classical models

The quantum computing community generally agree that any useful application of quantum computers will require both improvements in algorithm design to decrease the resource requirements and improvements in hardware design to increase the resources available to the algorithm. Additionally, the intersection of the lines 'the resources required by a useful algorithm' and 'the resources available on a quantum device', is the point quantum computing becomes practically applicable. Some predictions put this 10 years away, a cartoon, Figure 7.1, illustrates this concept.

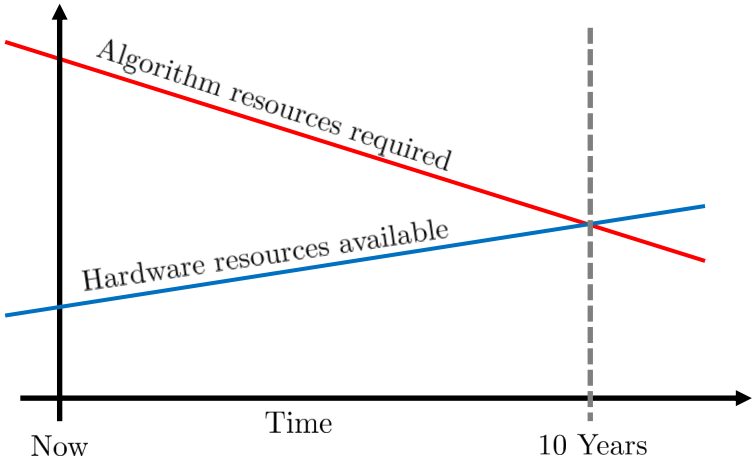


Figure 7.1: A cartoon illustrating the intersection of the lines 'the resources required by a useful algorithm' (red) and 'the resources available on a quantum device' (blue).

Algorithms

Problems in quantum chemistry are both part of the inspiration for quantum computers, [84, 159], and the potential first useful application [50, 168].

It is unlikely to be algorithms run on large scale fault-tolerant quantum devices such as quantum phase estimation [4], though attempts have been made to reduce these requirements [246]. Variational quantum algorithms have attracted a lot of attention for two reasons: They can be implemented in low-depth (not necessarily high performance) ways suitable for near-term hardware; and these algorithms are hybrid, which can play to the strengths of both quantum and classical computation.

State-of-the-art research in this area focuses on resource reduction and creating alternate algorithms that exploit the quantum properties of these devices, such as recent work on computing Green's function [78], simulation of the Hubbard model [48] and hardware implementations of imaginary time evolution [91].

Further, proposed theoretical quantum machine learning algorithms continue to show improvements over their classical counterparts [113], and more heuristic hybrid methods [170] are refining the allocation of quantum and classical resources most effectively. It is likely that the first useful application of quantum computers will be in one such hybrid system that may use neural networks in some way, for example in the gate-model [102], as described in Chapter 4, or quantum annealing [284], as described in Chapter 3, paradigms.

Hardware

There are many challenges facing the quantum computing community, no subgroup more so than those developing hardware. There are multitude companies and academic groups developing quantum computing hardware (including Google, IBM, D-Wave, Regetti, Honeywell, PsiQuantum) in various quantum computational frameworks, for example those studied in this Thesis, quantum annealing and gate-model.

However, there are unsolved questions about the scaling of these devices and minimizing errors resulting from decoherence [23], measurement, gate errors [280] and even loss (in the case of architectures based on photons) [223]. Certainly, large scale devices will require some form of error correction [87, 123].

Increasing the coherence time of these systems is especially important such that large depth algorithms can be run. For example, performance of QAOA improves with increased depth [20], where the accessible depth is related to the coherence time of the qubits in the circuit.

7.1.1 Classical models - quantum problems

Unlike quantum computers, neural networks have proven useful application. While there are fundamental challenges to understanding and characterizing neural networks [202, 206], these problems do not necessarily need to be solved in order to greatly improve the performance and decrease the cost of these algorithms.

Algorithms

There are three areas highlighted here for general neural network design and implementation, which may improve the efficiency. Firstly, it is possible to reduce the computational cost via pruning [86, 250], which roughly involves deleting parameters that are no longer considered useful. However realising these gains in efficiency will require innovations in GPU memory management. Secondly, mixed precision training can greatly decrease the cost of forward and backward passes of a neural network by replacing weights and data in parts of the network to half-precision floating point numbers [175, 297]. Finally, continued improvements on novel optimization methods, such as those used in this Thesis [96, 277], will promote fast and reliable optimization of the loss function, which currently often requires intuition, luck and repetition.

In the areas of modelling quantum states and quantum monte carlo, different theoretical Ansätze, such as Pfaffian wave functions [24] and geminal wave functions [56] might provide a better basis to develop neural network models. Further, neural networks are finding their place in other more efficient (though less accurate) methods such as Density Functional Theory [67], where the neural networks replace the electronic density functional.

Hardware

Neural networks are a more mature technology, having already experienced several false starts and AI winters in funding. As described in the introduction, the confluence of computational power, availability of data, model design and *lots of money* has launched neural networks from an academic curiosity to a fundamental part of business and increasingly a tool for the scientific research community to use in domain specific problems.

Transistor design, chip manufacturing and GPU architectures continue to improve, reducing the cost and time required to train large neural network models. Second, the quality and quantity of neural network algorithm research is forcing rapid and significant developments, both in the fundamentals of network design and optimization but also in domain specialisation. Finally, the infrastructure supporting neural networks is buoyed by an active and engaged community, such as popular frameworks such as Tensorflow and Pytorch.

Further into the future, alternate approaches to holding neural networks in memory [10] to neuromorphic chip designs, such as Loihi [68], and spiking neural networks [205] provide potential avenues to reducing the cost of running these algorithms well past the end of Moore's law.

7.1.2 Conclusion

I conclude that the path forward for quantum algorithms and quantum computational devices, specifically quantum machine learning, will be challenging and it is not clear what will be the first useful implementation of these devices. On the other side of the coin, neural networks, and their application to quantum mechanical problems, are already establishing state-of-the-art baselines in different areas. This coupled with the promise of improved hardware and methods make this a good research avenue for the advancement of classical neural network methods applied to quantum mechanical problems.

BIBLIOGRAPHY

[1] *Forest SDK*.
`https://github.com/rigetti`.

[2] *Ray*, 2020.

[3] S. AARONSON, *Read the fine print*, Nature Physics, 11 (2015), pp. 291–293.

[4] D. S. ABRAMS AND S. LLOYD, *Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors*, Physical Review Letters, 83 (1999), p. 5162.

[5] D. H. ACKLEY, G. E. HINTON, AND T. J. SEJNOWSKI, *A learning algorithm for Boltzmann machines*, in Readings in Computer Vision, Elsevier, 1987, pp. 522–533.

[6] S. H. ADACHI AND M. P. HENDERSON, *Application of quantum annealing to training of deep neural networks*, arXiv preprint arXiv:1510.06356, (2015).

[7] J. ADCOCK, E. ALLEN, M. DAY, S. FRICK, J. HINCHLIFF, M. JOHNSON, S. MORLEY-SHORT, S. PALLISTER, A. PRICE, AND S. STANISIC, *Advances in quantum machine learning*, arXiv preprint arXiv:1512.02900, (2015).

[8] T. ALBASH, S. BOIXO, D. A. LIDAR, AND P. ZANARDI, *Quantum adiabatic markovian master equations*, New Journal of Physics, 14 (2012), p. 123016.

[9] S.-I. AMARI, *Neural learning in structured parameter spaces-natural riemannian gradient*, in Advances in neural information processing systems, 1997, pp. 127–133.

[10] S. AMBROGIO, P. NARAYANAN, H. TSAI, R. M. SHELBY, I. BOYBAT, C. DI NOLFO, S. SIDLER, M. GIORDANO, M. BODINI, N. C. FARINHA, ET AL., *Equivalent-accuracy accelerated neural-network training using analogue memory*, Nature, 558 (2018), pp. 60–67.

[11] M. H. AMIN, *Searching for quantum speedup in quasistatic quantum annealers*, Physical Review A, 92 (2015), p. 052323.

[12] M. H. AMIN, E. ANDRIYASH, J. ROLFE, B. KULCHYTSKYY, AND R. MELKO, *Quantum Boltzmann machine*, arXiv preprint arXiv:1601.02036, (2016).

[13] J. B. ANDERSON, *Quantum chemistry by random walk*, The Journal of Chemical Physics, 65 (1976), pp. 4121–4127.

[14] M. ANDRYCHOWICZ, M. DENIL, S. GOMEZ, M. W. HOFFMAN, D. PFAU, T. SCHAUL, B. SHILLINGFORD, AND N. DE FREITAS, *Learning to learn by gradient descent by gradient descent*, in Advances in Neural Information Processing Systems, 2016, pp. 3981–3989.

[15] E. R. ANSCHUETZ AND C. ZANOCI, *Near-term quantum-classical associative adversarial networks*, arXiv preprint arXiv:1905.13205, (2019).

[16] B. APOLLONI, D. DE FALCO, AND N. CESA-BIANCHI, *A numerical implementation of" quantum annealing"*, tech. rep., 1988.

[17] N. ARGAMAN AND G. MAKOV, *Density functional theory: An introduction*, American Journal of Physics, 68 (2000), pp. 69–79.

[18] T. ARICI AND A. CELIKYILMAZ, *Associative adversarial networks*, arXiv preprint arXiv:1611.06953, (2016).

[19] M. ARJOVSKY, S. CHINTALA, AND L. BOTTOU, *Wasserstein gan*, arXiv preprint arXiv:1701.07875, (2017).

[20] F. ARUTE, K. ARYA, R. BABBUSH, D. BACON, J. C. BARDIN, R. BARENDS, S. BOIXO, M. BROUGHTON, B. B. BUCKLEY, D. A. BUELL, ET AL., *Quantum approximate optimization of non-planar graph problems on a planar superconducting processor*, arXiv preprint arXiv:2004.04197, (2020).

[21] G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, AND M. PROTASI, *Complexity and approximation: Combinatorial optimization problems and their approximability properties*, Springer Science & Business Media, 2012.

[22] J. BA, R. GROSSE, AND J. MARTENS, *Distributed second-order optimization using kronecker-factored approximations*, (2016).

[23] D. BACON, *Decoherence, control, and symmetry in quantum computers*, arXiv preprint quant-ph/0305025, (2003).

[24] M. BAJDICH, L. MITAS, G. DROBNỲ, L. WAGNER, AND K. SCHMIDT, *Pfaffian pairing wave functions in electronic-structure quantum monte carlo simulations*, Physical review letters, 96 (2006), p. 130201.

[25] R. M. BALABIN AND E. I. LOMAKINA, *Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies*, The journal of chemical physics, 131 (2009), p. 074104.

[26] S. A. BARNETT, *Convergence problems with generative adversarial networks (gans)*, arXiv preprint arXiv:1806.11382, (2018).

[27] D. BEASLEY, D. R. BULL, AND R. R. MARTIN, *An overview of genetic algorithms: Part 1, fundamentals*, University computing, 15 (1993), pp. 56–69.

[28] I. BELLO, B. ZOPH, V. VASUDEVAN, AND Q. V. LE, *Neural optimizer search with reinforcement learning*, in Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 459–468.

[29] M. BENEDETTI, J. R. GÓMEZ, AND A. PERDOMO-ORTIZ, *Quantum-assisted helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices*, Quantum Science and Technology, (2018).

[30] M. BENEDETTI, E. GRANT, L. WOSSNIG, AND S. SEVERINI, *Adversarial quantum circuit learning for pure state approximation*, arXiv preprint arXiv:1806.00463, (2018).

[31] M. BENEDETTI, E. LLOYD, AND S. SACK, *Parameterized quantum circuits as machine learning models*, arXiv preprint arXiv:1906.07682, (2019).

[32] M. BENEDETTI, J. REALPE-GÓMEZ, R. BISWAS, AND A. PERDOMO-ORTIZ, *Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning*, Physical Review A, 94 (2016), p. 022308.

[33] ——, *Quantum-assisted learning of hardware-embedded probabilistic graphical models*, Physical Review X, 7 (2017), p. 041052.

[34] Y. BENGIO, S. BENGIO, AND J. CLOUTIER, *Learning a synaptic learning rule*, Université de Montréal, Département d'informatique et de recherche opérationnelle, 1990.

[35] Y. BENGIO, P. SIMARD, P. FRASCONI, ET AL., *Learning long-term dependencies with gradient descent is difficult*, IEEE transactions on neural networks, 5 (1994), pp. 157–166.

[36] J. L. BERRAL, I. GOIRI, R. NOU, F. JULIA, J. GUITART, R. GAVALDA, AND J. TORRES, *Towards energy-aware scheduling in data centers using machine learning*, Proceedings of the 1st International Conference on energy-Efficient Computing and Networking, (2010), pp. 215–224.

[37] J. BIAMONTE, P. WITTEK, N. PANCOTTI, P. REBENTROST, N. WIEBE, AND S. LLOYD, *Quantum machine learning*, Nature, 549 (2017), pp. 195–202.

[38] C. H. BISCHOF, H. M. BÜCKER, P. HOVLAND, U. NAUMANN, AND J. UTKE, *Advances in automatic differentiation*, (2008).

[39] R. BISWAS, Z. JIANG, K. KECHEZHI, S. KNYSH, S. MANDRA, B. O'GORMAN, A. PERDOMO-ORTIZ, A. PETUKHOV, J. REALPE-GÓMEZ, E. RIEFFEL, ET AL., *A nasa perspective on quantum computing: Opportunities and challenges*, Parallel Computing, 64 (2017), pp. 81–98.

[40] M. BLENCOWE, *Quantum ram*, Nature, 468 (2010), pp. 44–45.

[41] S. F. BOYS AND N. C. HANDY, *A calculation for the energies and wavefunctions for states of neon with full electronic correlation accuracy*, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 310 (1969), pp. 63–78.

[42] B. BRANSDEN AND C. JOACHAIN, *Quantum mechanics,(2000)*, Prentice Hall.

[43] B. H. BRANSDEN AND C. J. JOACHAIN, *Introduction to quantum mechanics*, (1989).

[44] M. J. BREMNER, A. MONTANARO, AND D. J. SHEPHERD, *Average-case complexity versus approximate simulation of commuting quantum computations*, Physical review letters, 117 (2016), p. 080501.

[45] T. B. BROWN, B. MANN, N. RYDER, M. SUBBIAH, J. KAPLAN, P. DHARIWAL, A. NEELAKANTAN, P. SHYAM, G. SASTRY, A. ASKELL, ET AL., *Language models are few-shot learners*, arXiv preprint arXiv:2005.14165, (2020).

[46] M. P. BURRUUS, *Blog evaluating the cost of gpt-3*.

[47] R. H. BYRD, P. LU, J. NOCEDAL, AND C. ZHU, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on Scientific Computing, 16 (1995), pp. 1190–1208.

[48] J. CAI, W. G. MACREADY, AND A. ROY, *A practical heuristic for finding graph minors*, arXiv preprint arXiv:1406.2741, (2014).

[49] Z. CAI AND J. LIU, *Approximating quantum many-body wave functions using artificial neural networks*, Physical Review B, 97 (2018), p. 035116.

[50] Y. CAO, J. ROMERO, J. P. OLSON, M. DEGROOTE, P. D. JOHNSON, M. KIEFEROVÁ, I. D. KIVLICHAN, T. MENKE, B. PEROPADRE, N. P. SAWAYA, ET AL., *Quantum chemistry in the age of quantum computing*, Chemical reviews, 119 (2019), pp. 10856–10915.

[51] G. CARLEO AND M. TROYER, *Solving the quantum many-body problem with artificial neural networks*, Science, 355 (2017), pp. 602–606.

[52] J. CARRASQUILLA, *Machine learning for quantum matter*, arXiv preprint arXiv:2003.11040, (2020).

[53] J. CARRASQUILLA, G. TORLAI, R. G. MELKO, AND L. AOLITA, *Reconstructing quantum states with generative models*, Nature Machine Intelligence, 1 (2019), p. 155.

[54] M. A. CARREIRA-PERPINAN AND G. E. HINTON, *On contrastive divergence learning.*, in Aistats, vol. 10, Citeseer, 2005, pp. 33–40.

[55] M. CASULA, C. ATTACCALITE, AND S. SORELLA, *Correlated geminal wave function for molecules: An efficient resonating valence bond approach*, The Journal of chemical physics, 121 (2004), pp. 7110–7126.

[56] M. CASULA AND S. SORELLA, *Geminal wave functions with jastrow correlation: A first application to atoms*, The Journal of Chemical Physics, 119 (2003), pp. 6500–6511.

[57] D. M. CEPERLEY AND B. J. ALDER, *Ground state of the electron gas by a stochastic method*, Physical Review Letters, 45 (1980), p. 566.

[58] S. J. CHAKRAVORTY, S. R. GWALTNEY, E. R. DAVIDSON, F. A. PARPIA, AND C. F. P FISCHER, *Ground-state correlation energies for atomic ions with 3 to 18 electrons*, Physical Review A, 47 (1993), p. 3649.

[59] P. CHARBONNEAU, *An introduction to genetic algorithms for numerical optimization*, report, National Center for Atmospheric Research, 2002.

[60] Y. CHEN, M. W. HOFFMAN, S. G. COLMENAREJO, M. DENIL, T. P. LILLICRAP, M. BOTVINICK, AND N. DE FREITAS, *Learning to learn without gradient descent by gradient descent*, in Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 748–756.

[61] A. M. CHILDS AND W. VAN DAM, *Quantum algorithms for algebraic problems*, Reviews of Modern Physics, 82 (2010), p. 1.

[62] V. CHOI, *Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design*, Quantum Information Processing, 10 (2011), pp. 343–353.

[63] K. CHOO, G. CARLEO, N. REGNAULT, AND T. NEUPERT, *Symmetries and many-body excitations with neural-network quantum states*, Physical review letters, 121 (2018), p. 167204.

[64] K. CHOO, T. NEUPERT, AND G. CARLEO, *Study of the two-dimensional frustrated j1-j2 model with neural network quantum states*, arXiv preprint arXiv:1903.06713, (2019).

[65] C. CILIBERTO, M. HERBSTER, A. D. IALONGO, M. PONTIL, A. ROCCHETTO, S. SEVERINI, AND L. WOSSNIG, *Quantum machine learning: a classical perspective*, Proc. R. Soc. A, 474 (2018), p. 20170551.

[66] T. F. COVA AND A. A. PAIS, *Deep learning for deep chemistry: optimizing the prediction of chemical patterns*, Frontiers in Chemistry, 7 (2019).

[67] C. A. CUSTÓDIO, É. R. FILLETTI, AND V. V. FRANÇA, *Artificial neural networks for density-functional optimizations in fermionic systems*, Scientific reports, 9 (2019), pp. 1–7.

[68] M. DAVIES, N. SRINIVASA, T.-H. LIN, G. CHINYA, Y. CAO, S. H. CHODAY, G. DIMOU, P. JOSHI, N. IMAM, S. JAIN, ET AL., *Loihi: A neuromorphic manycore processor with on-chip learning*, IEEE Micro, 38 (2018), pp. 82–99.

[69] D.-L. DENG, X. LI, AND S. D. SARMA, *Quantum entanglement in neural network states*, Physical Review X, 7 (2017), p. 021021.

[70] L. DENG, D. YU, ET AL., *Deep learning: methods and applications*, Foundations and Trends® in Signal Processing, 7 (2014), pp. 197–387.

[71] C. DOERSCH, *Tutorial on variational autoencoders*, arXiv preprint arXiv:1606.05908, (2016).

[72] V. DOROBANTU, *The postulates of quantum mechanics*, arXiv preprint physics/0602145, (2006).

[73] P. O. DRAL, *Quantum chemistry in the age of machine learning*, The Journal of Physical Chemistry Letters, 11 (2020), pp. 2336–2347.

[74] N. DRUMMOND, Z. RADNAI, J. TRAIL, M. TOWLER, AND R. NEEDS, *Diffusion quantum monte carlo study of three-dimensional wigner crystals*, Physical Review B, 69 (2004), p. 085116.

[75] W. DUCH AND G. H. DIERCKSEN, *Neural networks as tools to solve problems in physics and chemistry*, Computer physics communications, 82 (1994), pp. 91–103.

[76] V. DUMOULIN, I. J. GOODFELLOW, A. C. COURVILLE, AND Y. BENGIO, *On the challenges of physical implementations of rbms.*, in AAAI, vol. 2014, 2014, pp. 1199–1205.

[77] V. DUNJKO, J. M. TAYLOR, AND H. J. BRIEGEL, *Quantum-enhanced machine learning*, Physical review letters, 117 (2016), p. 130501.

[78] S. ENDO, I. KURATA, AND Y. O. NAKAGAWA, *Calculation of the green's function on near-term quantum computers*, Physical Review Research, 2 (2020), p. 033281.

[79] E. FARHI, J. GOLDSTONE, AND S. GUTMANN, *A quantum approximate optimization algorithm*, arXiv preprint arXiv:1411.4028, (2014).

[80] E. FARHI, J. GOLDSTONE, S. GUTMANN, AND M. SIPSER, *Quantum computation by adiabatic evolution*, arXiv preprint quant-ph/0001106, (2000).

[81] E. FARHI AND H. NEVEN, *Classification with quantum neural networks on near term processors*, arXiv preprint arXiv:1802.06002, (2018).

[82] M. FEURER, A. KLEIN, K. EGGENSPERGER, J. SPRINGENBERG, M. BLUM, AND F. HUTTER, *Efficient and robust automated machine learning*, in Advances in neural information processing systems, 2015, pp. 2962–2970.

[83] R. FEYNMAN AND M. COHEN, *Energy spectrum of the excitations in liquid helium*, Physical Review, 102 (1956), p. 1189.

[84] R. P. FEYNMAN, *Simulating physics with computers*, Int. J. Theor. Phys, 21 (1982).

[85] W. FOULKES, L. MITAS, R. NEEDS, AND G. RAJAGOPAL, *Quantum monte carlo simulations of solids*, Reviews of Modern Physics, 73 (2001), p. 33.

[86] J. FRANKLE, G. K. DZIUGAITE, D. M. ROY, AND M. CARBIN, *Pruning neural networks at initialization: Why are we missing the mark?*, arXiv preprint arXiv:2009.08576, (2020).

[87] F. GAITAN, *Quantum error correction and fault tolerant quantum computing*, CRC Press, 2008.

[88] V. GIOVANNETTI, S. LLOYD, AND L. MACCONE, *Quantum random access memory*, Physical review letters, 100 (2008), p. 160501.

[89] G. GIULIANI AND G. VIGNALE, *Quantum theory of the electron liquid*, Cambridge university press, 2005.

[90] G. B. GOH, N. O. HODAS, AND A. VISHNU, *Deep learning for computational chemistry*, Journal of computational chemistry, 38 (2017), pp. 1291–1307.

[91] N. GOMES, F. ZHANG, N. F. BERTHUSEN, C.-Z. WANG, K.-M. HO, P. P. ORTH, AND Y. YAO, *Efficient step-merged quantum imaginary time evolution algorithm for quantum chemistry*, Journal of Chemical Theory and Computation, 16 (2020), pp. 6256–6266.

[92] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, MIT Press, 2016.

[93] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial nets*, in Advances in neural information processing systems, 2014, pp. 2672–2680.

[94] E. GRANT, L. WOSSNIG, M. OSTASZEWSKI, AND M. BENEDETTI, *An initialization strategy for addressing barren plateaus in parametrized quantum circuits*, arXiv:1903.05076, (2019).

[95] D. J. GRIFFITHS AND D. F. SCHROETER, *Introduction to quantum mechanics*, Cambridge University Press, 2018.

[96] R. GROSSE AND J. MARTENS, *A kronecker-factored approximate fisher matrix for convolution layers*, in International Conference on Machine Learning, 2016, pp. 573–582.

[97] L. K. GROVER, *Quantum mechanics helps in searching for a needle in a haystack*, Physical review letters, 79 (1997), p. 325.

[98] G. G. GUERRESCHI AND A. MATSUURA, *QAOA for Max-Cut requires hundreds of qubits for quantum speed-up*, Scientific reports, 9 (2019), p. 6903.

[99] G. G. GUERRESCHI AND M. SMELYANSKIY, *Practical optimization for hybrid quantum-classical algorithms*, arXiv:1701.01450, (2017).

[100] I. GULRAJANI, F. AHMED, M. ARJOVSKY, V. DUMOULIN, AND A. C. COURVILLE, *Improved training of wasserstein gans*, in Advances in Neural Information Processing Systems, 2017, pp. 5769–5779.

[101] L. GUPTA AND I. HEN, *Elucidating the interplay between non-stoquasticity and the sign problem*, Advanced Quantum Technologies, 3 (2020), p. 1900108.

[102] L. GYONGYOSI, *Unsupervised quantum gate control for gate-model quantum computers*, Scientific reports, 10 (2020), pp. 1–16.

[103] S. HADFIELD, *On the representation of Boolean and real functions as Hamiltonians for quantum computing*, arXiv:1804.09130, (2018).

[104] S. HADFIELD, Z. WANG, B. O'GORMAN, E. G. RIEFFEL, D. VENTURELLI, AND R. BISWAS, *From the quantum approximate optimization algorithm to a quantum alternating operator ansatz*, Algorithms, 12 (2019), p. 34.

[105] S. HADFIELD, Z. WANG, E. G. RIEFFEL, B. O'GORMAN, D. VENTURELLI, AND R. BISWAS, *Quantum approximate optimization with hard and soft constraints*, in Proceedings of the Second International Workshop on Post Moores Era Supercomputing, ACM, 2017, pp. 15–21.

[106] B. L. HAMMOND, W. A. LESTER, AND P. J. REYNOLDS, *Monte Carlo methods in ab initio quantum chemistry*, vol. 1, World Scientific, 1994.

[107] C. HEMPEL, C. MAIER, J. ROMERO, J. MCCLEAN, T. MONZ, H. SHEN, P. JURCEVIC, B. P. LANYON, P. LOVE, R. BABBUSH, ET AL., *Quantum chemistry calculations on a trapped-ion quantum simulator*, Physical Review X, 8 (2018), p. 031022.

[108] J. HERMANN, Z. SCHÄTZLE, AND F. NOÉ, *Deep-neural-network solution of the electronic schrödinger equation*, Nature Chemistry, 12 (2020), pp. 891–897.

[109] S. HOCHREITER, *The vanishing gradient problem during learning recurrent neural nets and problem solutions*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6 (1998), pp. 107–116.

[110] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[111] M. HOLZMANN, D. M. CEPERLEY, C. PIERLEONI, AND K. ESLER, *Backflow correlations for the electron gas and metallic hydrogen*, Physical Review E, 68 (2003), p. 046707.

[112] E. W. HUANG, C. B. MENDL, S. LIU, S. JOHNSTON, H.-C. JIANG, B. MORITZ, AND T. P. DEVEREAUX, *Numerical evidence of fluctuating stripes in the normal state of high-tc cuprate superconductors*, Science, 358 (2017), pp. 1161–1164.

[113] H.-Y. HUANG, R. KUENG, AND J. PRESKILL, *Information-theoretic bounds on quantum advantage in machine learning*, arXiv preprint arXiv:2101.02464, (2021).

[114] J. HUBBARD, *Electron correlations in narrow energy bands*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 276 (1963), pp. 238–257.

[115] P. ISOLA, J.-Y. ZHU, T. ZHOU, AND A. A. EFROS, *Image-to-image translation with conditional adversarial networks*, arXiv preprint, (2017).

[116] R. JASTROW, *Many-body problem with strong forces*, Physical Review, 98 (1955), p. 1479.

[117] Z.-A. JIA, B. YI, R. ZHAI, Y.-C. WU, G.-C. GUO, AND G.-P. GUO, *Quantum neural network states: A brief review of methods and applications*, Advanced Quantum Technologies, (2019).

[118] E. JONES, T. OLIPHANT, P. PETERSON, ET AL., *SciPy: Open source scientific tools for Python*, 2001-.

[119] M. I. JORDAN AND T. M. MITCHELL, *Machine learning: Trends, perspectives, and prospects*, Science, 349 (2015), pp. 255–260.

[120] P. JORDAN AND E. WIGNER, *über das Paulische äquivalenzverbot*, Z. Phys., 47 (1928), p. 631.

[121] J. JUMPER, R. EVANS, A. PRITZEL, T. GREEN, M. FIGURNOV, K. TUNYASUVUNAKOOL, O. RONNEBERGER, R. BATES, A. ZIDEK, A. BRIDGLAND, ET AL., *High accuracy protein structure prediction using deep learning*, Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book), 22 (2020), p. 24.

[122] A. KANDALA, A. MEZZACAPO, K. TEMME, M. TAKITA, M. BRINK, J. M. CHOW, AND J. M. GAMBETTA, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature, 549 (2017), pp. 242–246.

[123] A. KANDALA, K. TEMME, A. D. CÓRCOLES, A. MEZZACAPO, J. M. CHOW, AND J. M. GAMBETTA, *Error mitigation extends the computational reach of a noisy quantum processor*, Nature, 567 (2019), pp. 491–495.

[124] H. J. KAPPEN, *Learning quantum models from quantum or classical data*, arXiv preprint arXiv:1803.11278, (2018).

[125] T. KATO, *On the eigenfunctions of many-particle systems in quantum mechanics*, Communications on Pure and Applied Mathematics, 10 (1957), pp. 151–177.

[126] H. G. KATZGRABER, F. HAMZE, AND R. S. ANDRIST, *Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines*, Physical Review X, 4 (2014), p. 021008.

[127] M. J. KEARNS, *The computational complexity of machine learning*, MIT press, 1990.

[128] J. KESSLER, F. CALCAVECCHIA, AND T. D. KÜHNE, *Artificial neural networks as trial wave functions for quantum monte carlo*, arXiv preprint arXiv:1904.10251, (2019).

[129] A. KHOSHAMAN, W. VINCI, B. DENIS, E. ANDRIYASH, AND M. H. AMIN, *Quantum variational autoencoder*, arXiv preprint arXiv:1802.05779, (2018).

[130] J. KING, S. YARKONI, M. M. NEVISI, J. P. HILTON, AND C. C. MCGEOCH, *Benchmarking a quantum annealing processor with the time-to-target metric*, arXiv preprint arXiv:1508.05087, (2015).

[131] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv:1412.6980, (2014).

[132] A. Y. KITAEV, A. SHEN, M. N. VYALYI, AND M. N. VYALYI, *Classical and quantum computation*, no. 47, American Mathematical Soc., 2002.

[133] E. KNILL, G. ORTIZ, AND R. D. SOMMA, *Optimal quantum measurements of expectation values of observables*, Physical Review A, 75 (2007), p. 012328.

[134] D. KOLLER, N. FRIEDMAN, L. GETOOR, AND B. TASKAR, *Graphical models in a nutshell*, Introduction to statistical relational learning, (2007), pp. 13–55.

[135] I. KOSZTIN, B. FABER, AND K. SCHULTEN, *Introduction to the diffusion monte carlo method*, American Journal of Physics, 64 (1996), pp. 633–644.

[136] P. KRANTZ, M. KJAERGAARD, F. YAN, T. P. ORLANDO, S. GUSTAVSSON, AND W. D. OLIVER, *A quantum engineer's guide to superconducting qubits*, Applied Physics Reviews, 6 (2019), p. 021318.

[137] R. KRAUTHGAMER AND U. FEIGE, *A polylogarithmic approximation of the minimum bisection*, SIAM review, 48 (2006), pp. 99–130.

[138] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[139] H. G. KÜMMEL, *A biography of the coupled cluster method*, International Journal of Modern Physics B, 17 (2003), pp. 5311–5325.

[140] Q. A. I. LAB, *Google quantum artificial intelligence blogpost*.

[141] O. R. N. LABORATORY, *Summit*.

[142] I. E. LAGARIS, A. LIKAS, AND D. I. FOTIADIS, *Artificial neural network methods in quantum mechanics*, Computer Physics Communications, 104 (1997), pp. 1–14.

[143] N. LE ROUX AND Y. BENGIO, *Representational power of restricted Boltzmann machines and deep belief networks*, Neural computation, 20 (2008), pp. 1631–1649.

[144] J. LEBLANC, A. E. ANTIPOV, F. BECCA, I. W. BULIK, G. K.-L. CHAN, C.-M. CHUNG, Y. DENG, M. FERRERO, T. M. HENDERSON, AND C. A. JIMÉNEZ-HOYOS, *Solutions of the two-dimensional hubbard model: benchmarks and results from a wide range of numerical algorithms*, Phys. Rev. X, 5 (2015), p. 041041.

[145] Y. LECUN, Y. BENGIO, AND G. HINTON, *Deep learning*, nature, 521 (2015), p. 436.

[146] C. LEDIG, L. THEIS, F. HUSZÁR, J. CABALLERO, A. CUNNINGHAM, A. ACOSTA, A. AITKEN, A. TEJANI, J. TOTZ, Z. WANG, ET AL., *Photo-realistic single image super-resolution using a generative adversarial network*, arXiv preprint, (2016).

[147] M. A. LEE, K. SCHMIDT, M. KALOS, AND G. CHESTER, *Green's function monte carlo method for liquid he 3*, Physical Review Letters, 46 (1981), p. 728.

[148] C. LEMKE, M. BUDKA, AND B. GABRYS, *Metalearning: a survey of trends and technologies*, Artificial intelligence review, 44 (2015), pp. 117–130.

[149] Y. LEVINE, D. YAKIRA, N. COHEN, AND A. SHASHUA, *Deep learning and quantum entanglement: Fundamental connections with implications to network design*, arXiv preprint arXiv:1704.01552, (2017).

[150] K. LI AND J. MALIK, *Learning to optimize*, arXiv:1606.01885, (2016).

[151] X. LIANG, W.-Y. LIU, P.-Z. LIN, G.-C. GUO, Y.-S. ZHANG, AND L. HE, *Solving frustrated quantum many-particle models with convolutional neural networks*, Physical Review B, 98 (2018), p. 104426.

[152] M. W. LIBBRECHT AND W. S. NOBLE, *Machine learning applications in genetics and genomics*, Nature Reviews Genetics, 16 (2015), p. 321.

[153] H. W. LIN, M. TEGMARK, AND D. ROLNICK, *Why does deep and cheap learning work so well?*, Journal of Statistical Physics, 168 (2017), pp. 1223–1247.

[154] J.-G. LIU, S.-H. LI, AND L. WANG, *Lecture note on deep learning and quantum many-body computation*, (2018).

[155] S. LLOYD, *Quantum algorithm for solving linear systems of equations*, APS, 2010 (2010), pp. D4–002.

[156] A. LUND, M. J. BREMNER, AND T. RALPH, *Quantum sampling problems, bosonsampling and quantum supremacy*, npj Quantum Information, 3 (2017), p. 15.

[157] D. LUO AND B. K. CLARK, *Backflow transformations via neural networks for quantum many-body wave functions*, Physical Review Letters, 122 (2019), p. 226401.

[158] P. MALDONADO, A. SARSA, E. BUENDÍA, AND F. GÁLVEZ, *Quantum monte carlo ground state energies for the singly charged ions from li through ar*, The Journal of chemical physics, 133 (2010), p. 064102.

[159] Y. MANIN, *Computable and uncomputable*, Sovetskoye Radio, Moscow, 128 (1980).

[160] J. MARSHALL, A. DI GIOACCHINO, AND E. G. RIEFFEL, *Perils of embedding for sampling problems*, Physical Review Research, 2 (2020), p. 023020.

[161] J. MARSHALL, E. G. RIEFFEL, AND I. HEN, *Thermalization, freeze-out, and noise: Deciphering experimental quantum annealers*, Physical Review Applied, 8 (2017), p. 064025.

[162] J. MARSHALL, D. VENTURELLI, I. HEN, AND E. G. RIEFFEL, *The power of pausing: advancing understanding of thermalization in experimental quantum annealers*, arXiv preprint arXiv:1810.05881, (2018).

[163] J. MARSHALL, F. WUDARSKI, S. HADFIELD, AND T. HOGG, *Characterizing local noise in qaoa circuits*, 2020.

[164] J. MARTENS, *Publically available kfac implementation*.

[165] J. MARTENS, J. BA, AND M. JOHNSON, *Kronecker-factored curvature approximations for recurrent neural networks*, in International Conference on Learning Representations, 2018.

[166] J. MARTENS AND R. GROSSE, *Optimizing neural networks with kronecker-factored approximate curvature*, in International conference on machine learning, 2015, pp. 2408–2417.

[167] MC-AI, *blogpost on gpu computational power*.

[168] S. MCARDLE, S. ENDO, A. ASPURU-GUZIK, S. C. BENJAMIN, AND X. YUAN, *Quantum computational chemistry*, Reviews of Modern Physics, 92 (2020), p. 015003.

[169] J. R. MCCLEAN, S. BOIXO, V. N. SMELYANSKIY, R. BABBUSH, AND H. NEVEN, *Barren plateaus in quantum neural network training landscapes*, Nature communications, 9 (2018), p. 4812.

[170] J. R. MCCLEAN, J. ROMERO, R. BABBUSH, AND A. ASPURU-GUZIK, *The theory of variational hybrid quantum-classical algorithms*, New Journal of Physics, 18 (2016), p. 023023.

[171] W. S. MCCULLOCH AND W. PITTS, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115–133.

[172] W. L. MCMILLAN, *Ground state of liquid he 4*, Physical Review, 138 (1965), p. A442.

[173] P. MEHTA, M. BUKOV, C.-H. WANG, A. G. DAY, C. RICHARDSON, C. K. FISHER, AND D. J. SCHWAB, *A high-bias, low-variance introduction to machine learning for physicists*, Physics Reports, (2019).

[174] R. G. MELKO, G. CARLEO, J. CARRASQUILLA, AND J. I. CIRAC, *Restricted boltzmann machines in quantum physics*, Nature Physics, (2019), p. 1.

[175] P. MICIKEVICIUS, S. NARANG, J. ALBEN, G. DIAMOS, E. ELSEN, D. GARCIA, B. GINSBURG, M. HOUSTON, O. KUCHAIEV, G. VENKATESH, ET AL., *Mixed precision training*, arXiv preprint arXiv:1710.03740, (2017).

[176] MICROSOFT, *Blog on model sizes*.

[177] K. MILLS, M. SPANNER, AND I. TAMBLYN, *Deep learning and the schrödinger equation*, Physical Review A, 96 (2017), p. 042113.

[178] S. MOHAMED AND B. LAKSHMINARAYANAN, *Learning in implicit generative models*, arXiv preprint arXiv:1610.03483, (2016).

[179] M. MOHSENI, P. READ, H. NEVEN, S. BOIXO, V. DENCHEV, R. BABBUSH, A. FOWLER, V. SMELYANSKIY, AND J. MARTINIS, *Commercialize quantum technologies in five years*, Nature News, 543 (2017), p. 171.

[180] N. MOLL, P. BARKOUTSOS, L. S. BISHOP, J. M. CHOW, A. CROSS, D. J. EGGER, S. FILIPP, A. FUHRER, J. M. GAMBETTA, M. GANZHORN, ET AL., *Quantum optimization using variational algorithms on near-term quantum devices*, Quantum Science and Technology, 3 (2018), p. 030503.

[181] A. MONTANARO, *Quantum algorithms: an overview*, npj Quantum Information, 2 (2016), pp. 1–8.

[182] B. MOSELEY, M. OSBORNE, AND S. BENJAMIN, *Bayesian optimisation for variational quantum eigensolvers*.

[183] T. MUNKHDALAI AND H. YU, *Meta networks*, in Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2554–2563.

[184] A. NAGY AND V. SAVONA, *Variational quantum monte carlo method with a neural-network ansatz for open quantum systems*, Physical review letters, 122 (2019), p. 250501.

[185] G. NANNICINI, *Performance of hybrid quantum-classical variational heuristics for combinatorial optimization*, Physical Review E, 99 (2019), p. 013304.

[186] S. NATALE AND A. BALLATORE, *Imagining the thinking machine: Technological myths and the rise of artificial intelligence*, Convergence, 26 (2020), pp. 3–18.

[187] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, The computer journal, 7 (1965), pp. 308–313.

[188] A. NICHOL, J. ACHIAM, AND J. SCHULMAN, *On first-order meta-learning algorithms*, arXiv:1803.02999, (2018).

[189] M. A. NIELSEN AND I. CHUANG, *Quantum computation and quantum information*, 2002.

[190] M. Y. NIU, S. LU, AND I. L. CHUANG, *Optimizing QAOA: Success probability and runtime dependence on circuit depth*, arXiv:1905.12134, (2019).

[191] Y. NOMURA, A. S. DARMAWAN, Y. YAMAJI, AND M. IMADA, *Restricted boltzmann machine learning for solving strongly correlated quantum systems*, Physical Review B, 96 (2017), p. 205152.

[192] NVIDIA, *Nividia hardware specifications*.

[193] R. ORÚS, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics, 349 (2014), pp. 117–158.

[194] P. J. O'MALLEY, R. BABBUSH, I. D. KIVLICHAN, J. ROMERO, J. R. MCCLEAN, R. BARENDS, J. KELLY, P. ROUSHAN, A. TRANTER, N. DING, ET AL., *Scalable quantum simulation of molecular energies*, Physical Review X, 6 (2016), p. 031007.

[195] J. PALDUS AND X. LI, *A critical assessment of coupled cluster method in quantum chemistry*, Advances in Chemical Physics, 110 (1999), pp. 1–175.

[196] C. PAPADIMITRIOU, *Computational complexity*, Addison-Wesley, 1994.

[197] M. G. PARIS, *The modern tools of quantum mechanics*, The European Physical Journal Special Topics, 203 (2012), pp. 61–86.

[198] C.-Y. PARK AND M. J. KASTORYANO, *Geometry of learning neural quantum states*, Physical Review Research, 2 (2020), p. 023232.

[199] L. PAULING AND E. B. WILSON, *Introduction to quantum mechanics with applications to chemistry*, Courier Corporation, 2012.

[200] A. PERDOMO-ORTIZ, M. BENEDETTI, J. REALPE-GÓMEZ, AND R. BISWAS, *Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers*, arXiv preprint arXiv:1708.09757, (2017).

[201] A. PERUZZO, J. MCCLEAN, P. SHADBOLT, M.-H. YUNG, X.-Q. ZHOU, P. J. LOVE, A. ASPURU-GUZIK, AND J. L. O'BRIEN, *A variational eigenvalue solver on a photonic quantum processor*, Nature communications, 5 (2014), p. 4213.

[202] P. PETERSEN, *Neural network theory*, (2020).

[203] D. PFAU.
private communication, 2020.

[204] D. PFAU, J. S. SPENCER, A. G. MATTHEWS, AND W. M. C. FOULKES, *Ab initio solution of the many-electron schrödinger equation with deep neural networks*, Physical Review Research, 2 (2020), p. 033429.

[205] M. PFEIFFER AND T. PFEIL, *Deep learning with spiking neurons: opportunities and challenges*, Frontiers in neuroscience, 12 (2018), p. 774.

[206] T. POGGIO, A. BANBURSKI, AND Q. LIAO, *Theoretical issues in deep networks*, Proceedings of the National Academy of Sciences, (2020).

[207] J. PRESKILL, *Quantum computing in the nisq era and beyond*, Quantum, 2 (2018), p. 79.

[208] J. J. QUINN AND J. J. QUINN, *Pseudopotentials, correlations, and hierarchy states in quantum hall systems: When the composite fermion picture works and why*, Solid state communications, 140 (2006), pp. 52–60.

[209] A. RADFORD, L. METZ, AND S. CHINTALA, *Unsupervised representation learning with deep convolutional generative adversarial networks*, arXiv preprint arXiv:1511.06434, (2015).

[210] A. I. RAE AND J. NAPOLITANO, *Quantum mechanics*, (2015).

[211] R. RAUSSENDORF, *Measurement-based quantum computation with cluster states*, International Journal of Quantum Information, 7 (2009), pp. 1053–1203.

[212] S. RAVI AND H. LAROCHELLE, *Optimization as a model for few-shot learning*, (2016).

[213] J. RAYMOND, S. YARKONI, AND E. ANDRIYASH, *Global warming: Temperature estimation in annealers*, Frontiers in ICT, 3 (2016), p. 23.

[214] P. REBENTROST, M. MOHSENI, AND S. LLOYD, *Quantum support vector machine for big data classification*, Physical review letters, 113 (2014), p. 130503.

[215] E. G. RIEFFEL, S. HADFIELD, T. HOGG, S. MANDRÀ, J. MARSHALL, G. MOSSI, B. O'GORMAN, E. PLAMADEALA, N. M. TUBMAN, D. VENTURELLI, ET AL., *From ansätze to Z-gates: A NASA view of quantum computing*, arXiv:1905.02860, (2019).

[216] E. G. RIEFFEL AND W. H. POLAK, *Quantum computing: A gentle introduction*, MIT Press, 2011.

[217] A. ROCCHETTO, E. GRANT, S. STRELCHUK, G. CARLEO, AND S. SEVERINI, *Learning hard quantum distributions with variational autoencoders*, npj Quantum Information, 4 (2018), p. 28.

[218] J. ROMERO, R. BABBUSH, J. R. MCCLEAN, C. HEMPEL, P. J. LOVE, AND A. ASPURU-GUZIK, *Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz*, Quantum Science and Technology, 4 (2018), p. 014008.

[219] T. F. RØNNOW, Z. WANG, J. JOB, S. BOIXO, S. V. ISAKOV, D. WECKER, J. M. MARTINIS, D. A. LIDAR, AND M. TROYER, *Defining and detecting quantum speedup*, Science, 345 (2014), pp. 420–424.

[220] C. C. J. ROOTHAAN, *New developments in molecular orbital theory*, Reviews of modern physics, 23 (1951), p. 69.

[221] I. ROSS, *Calculations of the energy levels of acetylene by the method of antisymmetric molecular orbitals, including σ-π interaction*, Transactions of the Faraday Society, 48 (1952), pp. 973–991.

[222] N. C. RUBIN, *A hybrid classical/quantum approach for large-scale studies of quantum systems with density matrix embedding theory*, arXiv:1610.06910, (2016).

[223] T. RUDOLPH, *Why i am optimistic about the silicon-photonic route to quantum computing*, APL Photonics, 2 (2017), p. 030901.

[224] M. RUGGERI, S. MORONI, AND M. HOLZMANN, *Nonlinear network description for many-body quantum systems in continuous space*, Physical review letters, 120 (2018), p. 205302.

[225] T. SALIMANS, I. GOODFELLOW, W. ZAREMBA, V. CHEUNG, A. RADFORD, AND X. CHEN, *Improved techniques for training gans*, in Advances in Neural Information Processing Systems, 2016, pp. 2234–2242.

[226] T. SALIMANS, J. HO, X. CHEN, S. SIDOR, AND I. SUTSKEVER, *Evolution strategies as a scalable alternative to reinforcement learning*, (2017).

[227] A. SANTORO, S. BARTUNOV, M. BOTVINICK, D. WIERSTRA, AND T. LILLICRAP, *Meta-learning with memory-augmented neural networks*, in International conference on machine learning, 2016, pp. 1842–1850.

[228] K. SCHMIDT AND J. MOSKOWITZ, *Correlated monte carlo wave functions for the atoms he through ne*, The Journal of chemical physics, 93 (1990), pp. 4172–4178.

[229] M. SCHULD, V. BERGHOLM, C. GOGOLIN, J. IZAAC, AND N. KILLORAN, *Evaluating analytic gradients on quantum hardware*, Physical Review A, 99 (2019), p. 032331.

[230] M. SCHULD, M. FINGERHUTH, AND F. PETRUCCIONE, *Implementing a distance-based classifier with a quantum interference circuit*, arXiv preprint arXiv:1703.10793, (2017).

[231] H. J. SCHULZ, *Interacting fermions in one dimension: from weak to strong correlation*, arXiv e-prints, (1993), pp. cond–mat/9302006.

[232] K. SCHÜTT, M. GASTEGGER, A. TKATCHENKO, K.-R. MÜLLER, AND R. J. MAURER, *Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions*, Nature communications, 10 (2019), pp. 1–10.

[233] K. SCHÜTT, P.-J. KINDERMANS, H. E. S. FELIX, S. CHMIELA, A. TKATCHENKO, AND K.-R. MÜLLER, *Schnet: A continuous-filter convolutional neural network for modeling quantum interactions*, in Advances in neural information processing systems, 2017, pp. 991–1001.

[234] D. SEHAYEK, A. GOLUBEVA, M. S. ALBERGO, B. KULCHYTSKYY, G. TORLAI, AND R. G. MELKO, *Learnability scaling of quantum states: restricted boltzmann machines*, Physical Review B, 100 (2019), p. 195125.

[235] P. SETH, P. L. RÍOS, AND R. NEEDS, *Quantum Monte Carlo study of the first-row atoms and ions*, The Journal of chemical physics, 134 (2011), p. 084105.

[236] Y. SHINGU, Y. SEKI, S. WATABE, S. ENDO, Y. MATSUZAKI, S. KAWABATA, T. NIKUNI, AND H. HAKOSHIMA, *Boltzmann machine learning with a variational quantum algorithm*, arXiv preprint arXiv:2007.00876, (2020).

[237] P. W. SHOR, *Scheme for reducing decoherence in quantum computer memory*, Physical review A, 52 (1995), p. R2493.

[238] ——, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM review, 41 (1999), pp. 303–332.

[239] D. SILVER, A. HUANG, C. J. MADDISON, A. GUEZ, L. SIFRE, G. VAN DEN DRIESSCHE, J. SCHRITTWIESER, I. ANTONOGLOU, V. PANNEERSHELVAM, M. LANCTOT, ET AL., *Mastering the game of go with deep neural networks and tree search*, nature, 529 (2016), pp. 484–489.

[240] T. R. SOCIETY, *Machine learning: the power and promise of computers that learn by example*, The Royal Society, (2017).

[241] S. SORELLA, *Green function monte carlo with stochastic reconfiguration*, Physical review letters, 80 (1998), p. 4558.

[242] ——, *Generalized lanczos algorithm for variational quantum monte carlo*, Physical Review B, 64 (2001), p. 024512.

[243] J. C. SPALL ET AL., *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE transactions on automatic control, 37 (1992), pp. 332–341.

[244] J. C. SPALL, S. D. HILL, AND D. R. STARK, *Theoretical framework for comparing several stochastic optimization approaches*, in Probabilistic and Randomized Methods for Design under Uncertainty, Springer, 2006, pp. 99–117.

[245] J. S. SPENCER, D. PFAU, A. BOTEV, AND W. FOULKES, *Better, faster fermionic neural networks*, arXiv preprint arXiv:2011.07125, (2020).

[246] N. H. STAIR, R. HUANG, AND F. A. EVANGELISTA, *A multireference quantum krylov algorithm for strongly correlated electrons*, Journal of Chemical Theory and Computation, 16 (2020), pp. 2236–2245.

[247] Q. SUN, T. C. BERKELBACH, N. S. BLUNT, G. H. BOOTH, S. GUO, Z. LI, J. LIU, J. D. MCCLAIN, E. R. SAYFUTYAROVA, S. SHARMA, S. WOUTERS, AND G. K. CHAN, *Pyscf: the python based simulations of chemistry framework*, 2017.

[248] D.-W. SYSTEMS, *An introduction to d-wave hardware 2014*.

[249] A. SZABO AND N. S. OSTLUND, *Modern quantum chemistry: introduction to advanced electronic structure theory*, Courier Corporation, 2012.

[250] H. TANAKA, D. KUNIN, D. L. YAMINS, AND S. GANGULI, *Pruning neural networks without any data by iteratively conserving synaptic flow*, arXiv preprint arXiv:2006.05467, (2020).

[251] H. THANH-TUNG, T. TRAN, AND S. VENKATESH, *On catastrophic forgetting and mode collapse in generative adversarial networks*, arXiv preprint arXiv:1807.04015, (2018).

[252] G. TORLAI, G. MAZZOLA, J. CARRASQUILLA, M. TROYER, R. MELKO, AND G. CARLEO, *Many-body quantum state tomography with neural networks*, arXiv preprint arXiv:1703.05334, (2017).

[253] G. TORLAI AND R. G. MELKO, *Machine learning quantum states in the nisq era*, arXiv preprint arXiv:1905.04312, (2019).

[254] H. TOUVRON, A. VEDALDI, M. DOUZE, AND H. JÉGOU, *Fixing the train-test resolution discrepancy: Fixefficientnet*, arXiv preprint arXiv:2003.08237, (2020).

[255] M. TROYER AND U.-J. WIESE, *Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations*, Physical review letters, 94 (2005), p. 170201.

[256] A. M. TURING, *On computable numbers, with an application to the entscheidungsproblem*, J. of Math, 58 (1936), p. 5.

[257] C. UMRIGAR, M. NIGHTINGALE, AND K. RUNGE, *A diffusion monte carlo algorithm with very small time-step errors*, The Journal of chemical physics, 99 (1993), pp. 2865–2890.

[258] T. VANDAL, E. KODRA, S. GANGULY, A. MICHAELIS, R. NEMANI, AND A. R. GANGULY, *Deepsd: Generating high resolution climate change projections through single image super-resolution*, in Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining, ACM, 2017, pp. 1663–1672.

[259] VARIOUS, *DeepQMC*, 2020.

[260] G. VERDON, M. BROUGHTON, AND J. BIAMONTE, *A quantum algorithm to train neural networks using low-depth circuits*, arXiv:1712.05304, (2017).

[261] G. VERDON, M. BROUGHTON, J. R. MCCLEAN, K. J. SUNG, R. BABBUSH, Z. JIANG, H. NEVEN, AND M. MOHSENI, *Learning to learn with quantum neural networks via classical neural networks*, arXiv:1907.05415, (2019).

[262] G. VERDON, J. MARKS, S. NANDA, S. LEICHENAUER, AND J. HIDARY, *Quantum hamiltonian-based models and the variational quantum thermalizer algorithm*, arXiv preprint arXiv:1910.02071, (2019).

[263] P. VIDNEROVÁ AND R. NERUDA, *Evolution strategies for deep neural network models design*, CEUR Workshop Proceedings, 1885 (2017), pp. 159–166.

[264] R. VILALTA AND Y. DRISSI, *A perspective view and survey of meta-learning*, Artificial intelligence review, 18 (2002), pp. 77–95.

[265] W. VINCI, L. BUFFONI, H. SADEGHI, A. KHOSHAMAN, E. ANDRIYASH, AND M. AMIN, *A path towards quantum advantage in training deep generative models with quantum annealers*, Machine Learning: Science and Technology, (2020).

[266] W. VINCI AND D. A. LIDAR, *Optimally stopped optimization*, Physical Review Applied, 6 (2016), p. 054016.

[267] O. A. VON LILIENFELD AND K. BURKE, *Retrospective on a decade of machine learning for chemical discovery*, Nature communications, 11 (2020), pp. 1–4.

[268] K. WANG, C. GOU, Y. DUAN, Y. LIN, X. ZHENG, AND F.-Y. WANG, *Generative adversarial networks: introduction and outlook*, IEEE/CAA Journal of Automatica Sinica, 4 (2017), pp. 588–598.

[269] Z. WANG, S. HADFIELD, Z. JIANG, AND E. G. RIEFFEL, *Quantum approximate optimization algorithm for MaxCut: A fermionic view*, Physical Review A, 97 (2018), p. 022304.

[270] Z. WANG, N. C. RUBIN, J. M. DOMINY, AND E. G. RIEFFEL, *XY-mixers: Analytical and numerical results for QAOA*, arXiv:1904.09314, (2019).

[271] B. WEATHER, *Historical temperatures in bristol (uk), dataset*.

[272] D. WECKER, M. B. HASTINGS, AND M. TROYER, *Progress towards practical quantum variational algorithms*, Physical Review A, 92 (2015), p. 042303.

[273] ——, *Training a quantum optimizer*, Physical Review A, 94 (2016), p. 022309.

[274] P. WERBOS, *Beyond regression:" new tools for prediction and analysis in the behavioral sciences*, Ph. D. dissertation, Harvard University, (1974).

[275] T. WESTERHOUT, N. ASTRAKHANTSEV, K. S. TIKHONOV, M. KATSNELSON, AND A. A. BAGROV, *Neural quantum states of frustrated magnets: generalization and sign structure*, arXiv preprint arXiv:1907.08186, (2019).

[276] T. WHITE, *Sampling generative networks: Notes on a few effective techniques*, arXiv preprint arXiv:1609.04468, (2016).

[277] O. WICHROWSKA, N. MAHESWARANATHAN, M. W. HOFFMAN, S. G. COLMENAREJO, M. DENIL, N. DE FREITAS, AND J. SOHL-DICKSTEIN, *Learned optimizers that scale and generalize*, in Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 3751–3760.

[278] N. WIEBE AND C. GRANADE, *Can small quantum systems learn?*, arXiv preprint arXiv:1512.03145, (2015).

[279] E. WIGNER, *On the interaction of electrons in metals*, Physical Review, 46 (1934), p. 1002.

[280] D. WILLSCH, M. NOCON, F. JIN, H. DE RAEDT, AND K. MICHIELSEN, *Gate-error analysis in simulations of quantum computers with transmon qubits*, Physical Review A, 96 (2017), p. 062302.

[281] M. WILSON, N. GAO, F. WUDARSKI, E. RIEFFEL, AND N. M. TUBMAN, *Simulations of state-of-the-art fermionic neural network wave functions with diffusion monte carlo*, arXiv preprint arXiv:2103.12570, (2021).

[282] M. WILSON, S. STROMSWOLD, F. WUDARSKI, S. HADFIELD, N. M. TUBMAN, AND E. RIEFFEL, *Optimizing quantum heuristics with meta-learning*, arXiv preprint arXiv:1908.03185, (2019).

[283] M. WILSON, T. VANDAL, T. HOGG, AND E. RIEFFEL, *Quantum-assisted associative adversarial network: Applying quantum annealing in deep learning*, arXiv preprint arXiv:1904.10573, (2019).

[284] W. WINCI, L. BUFFONI, H. SADEGHI, A. KHOSHAMAN, E. ANDRIYASH, AND M. H. AMIN, *A path towards quantum advantage in training deep generative models with quantum annealers*, Machine Learning: Science and Technology, 1 (2020), p. 045028.

[285] A. WOITZIK, *Entanglement in quantum-classical variational algorithms*, Master's thesis, Albert-Ludwigs-Universit ät Freiburgm Germany, 2018.

[286] A. J. C. WOITZIK, P. K. BARKOUTSOS, F. WUDARSKI, A. BUCHLEITNER, AND I. TAVERNELLI, *Entanglement production and convergence properties of the variational quantum eigensolver*, (2020).

[287] D. WU, L. WANG, AND P. ZHANG, *Solving statistical mechanics using variational autoregressive networks*, Physical review letters, 122 (2019), p. 080602.

[288] J. WU AND W. ZHANG, *Finding quantum many-body ground states with artificial neural network*, arXiv preprint arXiv:1906.11216, (2019).

[289] X. XU, Y. DING, S. X. HU, M. NIEMIER, J. CONG, Y. HU, AND Y. SHI, *Scaling for edge inference of deep neural networks*, Nature Electronics, 1 (2018), pp. 216–222.

[290] C. XUE, Z.-Y. CHEN, Y.-C. WU, AND G.-P. GUO, *Effects of quantum noise on quantum approximate optimization algorithm*, 2019.

[291] L. YANG, Z. LENG, G. YU, A. PATEL, W.-J. HU, AND H. PU, *Deep learning-enhanced variational monte carlo method for quantum many-body physics*, Physical Review Research, 2 (2020), p. 012039.

[292] W. YANG, L. PENG, Y. ZHU, AND L. HONG, *When machine learning meets multiscale modeling in chemical reactions*, arXiv preprint arXiv:2006.00700, (2020).

[293] F. YU, A. SEFF, Y. ZHANG, S. SONG, T. FUNKHOUSER, AND J. XIAO, *Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop*, arXiv preprint arXiv:1506.03365, (2015).

[294] R. ZEN, L. MY, R. TAN, F. HEBERT, M. GATTOBIGIO, C. MINIATURA, D. POLETTI, AND S. BRESSAN, *Transfer learning for scalability of neural-network quantum states*, arXiv preprint arXiv:1908.09883, (2019).

[295] S. ZHANG, L. YAO, A. SUN, AND Y. TAY, *Deep learning based recommender system: A survey and new perspectives*, ACM Computing Surveys (CSUR), 52 (2019), pp. 1–38.

[296] J. ZHAO, M. MATHIEU, AND Y. LECUN, *Energy-based generative adversarial network*, arXiv preprint arXiv:1609.03126, (2016).

[297] R. ZHAO, W. LUK, C. XIONG, X. NIU, AND K. H. TSOI, *On the challenges in programming mixed-precision deep neural networks*, in Proceedings of the 4th ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, 2020, pp. 20–28.

[298] C. ZOUFAL, A. LUCCHI, AND S. WOERNER, *Variational quantum boltzmann machines*, arXiv preprint arXiv:2006.06004, (2020).