Explorations in machine learning for interacting many-body systems

by

Jonathon Matthew Schulz-Beach

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Physics

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor:	Roger G. Melko Professor, Department of Physics, University of Waterloo Associate Faculty, Perimeter Institute for Theoretical Physics
Internal Member:	Anton Burkov Professor, Department of Physics, University of Waterloo
Internal Member:	Rob Myers Director, Perimeter Institute for Theoretical Physics Adjunct Professor, Department of Physics, University of Waterloo
Internal-External Member:	Pierre-Nicholas Roy Professor, Department of Chemistry, University of Waterloo
External Examiner:	Federico Becca Professor, University of Trieste, Department of Physics

Author's Declaration

Statement of Contributions

Chapter 2 consists of work previously published as [1] in collaboration with Anna Golubeva and Roger G. Melko. Chapter 2 also consists of a blog post [2] published by the author.

Chapter 3 consists of work previously published in [3] with co-authors Stavros Efthymiou and Roger G. Melko.

Chapter 5 consists of work previously published in [4] with co-authors Tarun Grover, Timothy H. Hsieh and Roger G. Melko.

Chapter 4 consists of work previously published in [5] with co-authors Isaac De Vlugt, Anna Golubeva, Patrick Huembeli, Bohdan Kulchytskyy, Xiuzhe Luo, Roger G Melko, Ejaaz Merali, Giacomo Torlai. Chapter 4 also consists of original material from collaborations with Michael Albergo, Florian Hopfmeuller and Ejaaz Merali.

Abstract

Most interacting many-body systems in physics are not analytically solvable. Instead, numerical methods are needed for the study of these complex and high-dimensional problems. At present, there are many interesting problems in strongly correlated systems that remain unsolved with current methods. At the heart of this problem is finding an efficient representation that incorporates symmetries, correlations and general features.

In the context of computer science, machine learning techniques have had astonishing success at reducing the dimensionality of data. The leading method is through the use of artificial neural networks. These networks have been enormously successful at sifting through vast amounts of data to find patterns and regularities. In a sense, neural networks are themselves a statistical system whose properties are adjusted to mimic the features of the data. By finding an effective low-dimensional representation of the data, machine learning has greatly subdued the curse of dimensionality found in many real-world problems.

In this Thesis, we apply several machine learning techniques to the study of interacting many-body systems in classical and quantum statistical physics. We explore supervised classification of phases of matter with an emphasis on physical interpretation of the network. In doing so, we design a custom network architecture that possesses rotational symmetry as an inductive bias. We further investigate connections between the renormalization group and deep learning through applying a super-resolving neural network to the classical Ising model. Towards experimental efforts, we also repurpose generative machine learning to quantum state tomography for the calibration and testing of quantum devices. We conclude with a latent variable model inspired by near-term quantum algorithms. This maps to a variational Monte Carlo ansatz that produces samples efficiently for interacting quantum systems.

Acknowledgements

There are many people without whom this Thesis would not be possible. First and foremost, I am thankful to my advisor Roger Melko. Roger provides a deep commitment to each student, treating them as capable researchers from the start, providing opportunities for growth, and above all respecting their questions and concerns. His patience, timely feedback, and guidance made this process much more enjoyable.

I was fortunate to have carried out my doctoral studies along with fellow students Lauren, Bohdan, Ponte, Giacomo, and Anna. I would not have learned nearly as much without the constant discussions and collaborations. A special thanks goes to Lauren for all the times she explained simple solutions to all my questions.

I am grateful to have collaborated with so many great physicists: Tim Hsieh, Juan Carrasquilla, Tarun Grover, Sebastian Wetzel, Estelle Inack, Emilie Huffman. It was also a pleasure to work with many enthusiastic and talented students: Ejaaz, Isaac, Florian, Mohamed, Michael, Stavros, Dan S., Dan K., Brian, Patrick, and Nam. Lastly, I thank my committee members, Anton Burkov, Rob Myers, and Pierre-Nicholas Roy for always providing helpful feedback and consistently asking important questions.

I would also like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for continued support. Additionally, I thank the University of Waterloo, the Perimeter Institute for Theoretical Physics, and the Vector Institute for Artificial Intelligence for accommodating me throughout my studies.

Finally, I am profoundly thankful to my parents for providing me with unwavering support throughout my studies. Without them, I would not have been fortunate enough to pursue a PhD in physics. Above all, I am deeply grateful to my wife, Rachael. I cannot imagine going through this without her constant love and support.

Dedication

To my wife, Rachael.

Table of Contents

List of Figures

List of Tables

1

Introduction

Physics largely assumes that the behaviour of matter and energy can be described by simple mathematical laws. Remarkably, these laws depend on the energy scale in question. At low energies, classical Newtonian mechanics is undisputedly successful. At higher energies, classical intuition gives way to quantum mechanical rules. For even higher energies, the standard model of particle physics becomes necessary to explain the interactions between ultra-small electrons, photons and other elementary fields.

The idea that all large-scale behaviours can be predicted from the most fundamental laws has a long history in science. Indeed, the reductionist viewpoint has been a driving force for particle physics, as we build increasingly large particle accelerators to probe increasingly high energies. And yet, there are many interesting phenomena presently unexplained by fundamental laws. These include high-temperature superconductivity, quantum spin liquids, amorphous solids, and many-body localization [6, 7, 8]. As Anderson pointed out in a famous essay, "*The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe*" [9]. This stands in stark contrast to Newton's clockwork universe.

Even in classical physics, a gas composed of at least $N > 10^{23}$ molecules poses a formidable challenge. There are $6N$ Newtonian equations of motion for the movement all particles. Solving these are beyond the capabilities of any computer. Rather than admit defeat, physicists recognize important properties of the system that the constituents themselves do not possess: namely, the macroscopic properties of temperature and pressure. In one sense, pressure and temperature are not fundamental since they can be derived from the kinetic theory of gases. Yet, it is hard to deny their usefulness in thermodynamics, let alone everyday experiences. This is one example where macroscopic properties can be derived from a microscopic system, and yet simultaneous take on a meaning of their own.

Interacting many-body systems pose an even more serious difficulty. With only 75 interacting binary degrees of freedom, we already face the same complexity as 10^{23} non-interacting particles. In the quantum setting, the non-relativist Schrödinger equation provides a framework to predict almost all properties of matter. However, the amount of computer memory required to store an entire many-body wavefunction scales exponentially in the number of qubits. This "catastrophe of dimension"[1] the simulation of even 265 particles, for which the amount of memory required is 2^{265} (approximately the number of elementary particles in the entire universe).

The curse of dimensionality is not only a problem for physicists. High-dimensional data is encountered in many areas, including human language, portfolio optimization, and natural images. In a standard high-definition image, there are 1920×1080 pixels, each with three-colour channels (RGB) that take on a range $0, \ldots, 255$. So the number of possible images (or microstates) is approximately 10^{10^7}. This number is already much greater than the estimated number of particles in the universe. Of course, most of these images are unrecognizable to humans, and many are duplicates (up to infinitesimal white noise). Only a small subset are so-called natural images that represent real photographs. What properties of an image make them "real"? How can we design a model to describe what a photograph of a dog is?

Designing an algorithm by hand for the 10^{10^7} microscopic pixels in an image is analogous to the interacting many-body problem in physics. The sheer scale and complexity prohibit any brute-force or lookup-table approach. Instead, we must rely on finding effective macroscopic properties in order to make progress.

There are broadly two approaches to solve high-dimensional problems. The first is stochastic sampling methods such as Monte Carlo, where given the microscopic system, we generate statistical measures using random sampling. Stochastic methods typically require knowledge about the microscopic model but are asymptotically exact. The second method is variational approximations. This encompasses finding a parameterized representation of the system. Such variational models can be considered compressions since they reduce the number of parameters from an exponential number to a manageable amount. Examples in physics include matrix product states [11], variational Slater-Jastrow wavefunctions [12], or neural networks [13]. In a sense, the variational approach gives a framework in which to *discover* to best model of underlying data. In Table 1.1, we compare common strengths and weaknesses of stochastic and variational approaches.

[1]More commonly called the *curse of dimensionality*, this phrase is from Laughlin and Pines' famous essay on emergence [10].

Variational Methods	**Stochastic sampling**
• biased	• asymptotically exact (unbiased)
• parametric model	• non-parametric model
• typically fast	• typically slow
• maximizing an explicit objective	• needs microscopic knowledge
• can be trapped in local minima	• can be trapped in local minima
• easy to verify convergence	• difficult to prove convergence

Table 1.1: Comparison of variational methods and stochastic methods.

Variational methods need to have sufficient representational power to approximate the true system. Many variational approximations are motivated by physical intuition[2], including mean-field theory and the Bethe ansatz [14, 15]. However, a relatively recent approach is to use incredibly high-dimensional systems, such as artificial neural networks, to mimic and reproduce the properties of a different system (i.e. natural images) [16]. Rather than enforcing our assumptions about the system a priori, we leverage emergence to our advantage. Laughlin and Pines called the science of the next century that of "complex adaptive matter", but instead we are in the era of large-scale *complex adaptive algorithms*: algorithms that learn from data.

The last decade has seen the rapid insurgence of machine learning tools adopted into everyday life. From search engines using state-of-the-art language models, to recommender systems on social-media, there is almost no technology that has not been influenced by the ground breaking improvement of machine learning models.

At the heart of machine learning is discovering patterns and structure in data. This is something very familiar to physicists. For instance, in the 16th century, Johannes Kepler discovered the laws of planetary motion from astronomical data collected by Tycho Brahe. Even Planck's solution to black-body radiation was motivated by fitting a theoretical model to the available data.

The proliferation of data in modern society coupled with increased computational power from using graphics processing units (GPUs) and dedicated hardware [17, 18] has created the widespread success of machine learning. In particular, artificial neural network models which were based on the biological neurons in the brain have repeatedly achieved state-of-the-art results on various benchmarks including computer vision, language translation, speech recognition, and games such as chess or Go.

[2]This is called inductive bias.

Machine learning has tamed the complexity of many real-world problems. We are overdue to integrate these methods back into physics where interacting systems still pose a serious challenge.

Outline

In this Thesis we explore various questions about using adaptive learning algorithms for the study of physical many-body systems. For the remainder of this Chapter, we provide the necessary background for understanding both stochastic Monte Carlo approaches, and the basics of machine learning with neural networks.

In Chapter 2, we use neural networks to classify phases separated by a Kosterlitz-Thouless transition in the two-dimensional classical XY model. We begin with standard supervised learning to distinguish the phases. Next, we pose the problem of identifying vortices directly with supervised learning. We proceed by using a semi-supervised "learning by confusion" method to identify the critical point. Lastly, we consider incorporating rotational symmetry into a network so that it makes learning internal representations of topological defects more natural.

In Chapter 3, we propose a supervised neural network to extrapolate finite-size systems to larger sizes. This is inspired by image super-resolution where high-definition images are upscaled to appear at 4k resolutions[3]. This idea of upscaling a small system to a large system draws many parallels with renormalization group (RG) procedures. Essentially, we train the network to invert RG decimation on the one- and two-dimensional Ising models.

In Chapter 4, we use a generative model called a restricted Boltzmann machine to approximate a quantum state from projective measurements of the state. We show how experimental or simulation data can be used to train the network to represent the state to high accuracy. As a proof of concept, we also demonstrate quantum state tomography with the modern transformer neural network. We highlight the improved performance granted by autoregressive sampling as compared to Monte Carlo sampling.

In Chapter 5, we design quantum-inspired variation ansatz to solve the quantum Ising model. The variational imaginary time ansatz is a latent variable model where the latent space is auxiliary degrees of freedom from the quantum-to-classical mapping. This can also be interpreted as a hybrid variational-worldline Monte Carlo method. We solve the one- and two-dimensional Ising models, as well as the one-dimensional non-integrable model

[3]This process occurs on any 4k television, but more advanced algorithms such as those using machine learning, can create a much more realistic and sharper image.

with a longitudinal field. Using both exact mappings to fermions and numerical Monte Carlo, we verify that the ansatz has a remarkable parameter efficiency.

In Chapter 6, we summarize the main findings in the Thesis and subsequent research developments. We conclude with an outlook for the interdisciplinary efforts of machine learning and many-body physics.

1.1 Stochastic sampling methods

Monte Carlo methods are perhaps one the most important numerical techniques of the last century. The key idea behind Monte Carlo tools is to use random sampling to approximate a solution to a problem that is infeasible otherwise. This profound, yet simple, idea has made Monte Carlo applicable to nearly every technical field; from financial engineering [19, 20], computational biology [21], particle physics [22], and even string theory [23].

In condensed matter and statistical physics, Monte Carlo offers a powerful tool for studying interacting systems. For classical systems, Monte Carlo is used for studying thermal properties of materials and phase transitions [24, 25, 26]. In the quantum realm, quantum Monte Carlo (QMC) is the leading method for systems in more than two spatial dimensions. QMC relies on mapping a d-dimensional quantum system to a $(d+1)$-dimensional classical system through an imaginary time path integral [27, 28]. Several variations of QMC exist for different types of systems: for highly-efficient stochastic series expansion for spin system [29], determinantal methods [30], and variational Monte Carlo (VMC) [31, 32]. All these methods depend on common ideas presented in this Section. These techniques all employ Markov chains and local updates. We conclude this Section with autoregressive sampling, a non-Markov chains method useful in machine learning models.

1.1.1 Monte Carlo for statistical physics

For many high-dimensional problems, Monte Carlo provides a way to approximate a solution up to a sampling error. These methods depend on drawing samples from a probability distribution, which could be useful on their own, or used to approximate a sum or integral.

In statistical mechanics, we are concerned with sampling microstates of a system following a Boltzmann distribution. For such a distribution, the probability of a state s with energy $E(s)$ at temperature $T = 1/\beta$ is

$$p(s) = \frac{e^{-\beta E(s)}}{Z} \tag{1.1}$$

where Z is a normalization constant called the partition function,

$$Z = \sum_{\{s\}} e^{-\beta E(s)} \tag{1.2}$$

and $\{s\}$ denotes all possible microstates.

We are interested in computing expectation values of observables in the canonical ensemble. Such examples include the magnetization, energy density, or specific heat. For an observable $\mathcal{O}$, the expectation value or thermal average is given by

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \sum_{\{s\}} \mathcal{O}(s)\, e^{-\beta E(s)} = \sum_{\{s\}} \mathcal{O}(s)\, p(s)\,. \tag{1.3}$$

For a many-body system, the sum is generally intractable because of an exponential number of possible configurations of a system.

The idea of Monte Carlo is to approximate the sum by randomly sampling the states that give the greatest contributions. Instead of weighting randomly chosen configurations with $p(s)$, we weight them evenly but choose them with probability $p(s)$. So for N states selected randomly according to $p(s_i)$, the expectation value is approximated by the empirical average

$$\langle \mathcal{O} \rangle = \sum_{\{s\}} \mathcal{O}(s)\, p(s) \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}(s_i)\,. \tag{1.4}$$

This sample mean is an *estimator* to the true expectation value. By the Central Limit Theorem, as the sample size tends to infinity ($N \to \infty$), the estimator converges to the expected value, with the caveat that the variance must remain finite. Moreover, if the samples are independent, the approximation is unbiased.

The error in the estimator over a sample of N independent and identically drawn (i.i.d.) samples is $O\left(\frac{1}{\sqrt{N}}\right)$. Consequently, the accuracy of Monte Carlo methods is limited. For a ten-fold increase in accuracy, we require a hundred-fold increase in the sample size. However, the statistical error is independent of the dimensionality of the integral, making Monte Carlo suitable for high-dimensional problems.

So far we have relied on sampling from $p(s)$, but this might be a hard problem itself. In the next Section, we introduce Markov chain Monte Carlo where we bypass the need to sample directly from $p(s)$ at the expense of having correlated samples. This technique is crucial to all data sets produced in this Thesis, as well as for machine learning with undirected graphical models.

1.1.2 Markov chain Monte Carlo

One method to generate samples from p is through use of Markov chain Monte Carlo (MCMC). A Markov chain is a sequence of states where each state only depends on the previous one. For a state x, we propose an update to a new state x' with transition probability $T(x \to x')$. The total probability of a state $\pi(x')$ is the sum of the probabilities from all other states to transition into x'

$$\pi(x') = \sum_x T(x \to x')\pi(x)\,. \tag{1.5}$$

This is necessary for the chain to model a stationary distribution.

The goal is that by designing an appropriate transition T, the chain probability $\pi(x)$ converges to the desired distribution $p(x)$. For this, we need to satisfy two conditions. The first is *ergodicity*: the conditions that any state x' can be reached from any other state x with a finite number of steps. This does not imply that $T(x \to x') \neq 0$, only that there must only exist a possible sequence $x \to x_1 \to \ldots \to x'$ with finite probability.

The second condition is called *balance*. The equation for balance is simply a solution to the stationarity condition in Eq. 1.5. This expresses that the total probability of reaching a state equals the total probability of leaving the state, i.e.

$$\sum_{x' \neq x} T(x' \to x_i)\pi(x_i') = \sum_{x \neq x'} T(x \to x')\pi(x)\,. \tag{1.6}$$

The simplest way to satisfy balance is by requiring local or *detailed balance*

$$T(x \to x')\,\pi(x) = T(x' \to x)\,\pi(x')\,. \tag{1.7}$$

A Markov chain that satisfies detailed balance is called *reversible*.

Starting from a random state, x_0, Markov chains repeatedly apply stochastic updates according to T to produce a sequence of states $x_1, \ldots, x_N$. The balance and ergodicity alone are not enough to guarantee the initial few states are representative of the stationary distribution. This is particularly true if the initial x_0 happens to be in a region of very small probability. A way to avoid this is by discarding the first n states in the chain. This is called *burning-in* or *equilibrating* the chain. This incurs an additional cost that can be prohibitive in some cases. One mitigation is to start the initial x_0 in a region of high-probability, although this may be difficult[4].

The second issue is that any two successive samples are correlated. In certain systems, the number of Markov steps required to produce roughly independent samples scales as a polynomial function of system size. This is *critical slowing down*, and makes MCMC very computationally expensive. For instance, at the critical point in the two-dimensional Ising model the autocorrelation time scales as $\tau \sim L^{2.2}$ [33] with local updates.

The final problem in MCMC is the possible breakdown of ergodicity, that is, sampling many disconnected regions of high probability. While the ergodicity condition ensures any state can in principle reach any other, it may take an exponential number of Markov steps to do so. If an update is not sufficiently global, it can easily result in the chain being trapped near a local high probability zone. A simple example is the low-temperature two-dimensional Ising model where at $T < T_c$, the phase space consists of two disjoint regions, all spins up and all spins down. In this case, a local update is not sufficient to escape a high-probability region of phase space. We call this ergodicity breaking, or the mode-mixing problem in MCMC. The use of multiple Markov chains, global updates or parallel tempering can help mitigate this issue [34, 35].

In the next Sections, we discuss three types of sampling algorithms: Metropolis-Hastings, Gibbs, and autoregressive sampling. All three will be important to the data generation used throughout this Thesis.

1.1.3 Metropolis-Hasting algorithm

In the 1950s, Metropolis, the Rosenbluth's, and the Teller's [36] were the first to use MCMC in physics. They considered designing the transition probability $T(x \to x')$ to be a small local change in a configuration. The algorithm was further generalized by Hastings in 1970 [37] who showed that we can generally decompose the transition matrix into an acceptance probability $A(x \to x')$, and a proposal probability $g(x \to x')$,

$$T(x \to x') = g(x \to x')A(x \to x') \,. \tag{1.8}$$

[4]Chapter 3 investigates using machine learning to find good starting points.

In the case of the Ising model, we take $g(x \to x')$ to be a single spin flip $s \to s'$. Assuming the chain has converged so that $\pi(x) = p(x)$, detailed balance for the Boltzmann distribution reduces to

$$\frac{g(s \to s')A(s \to s')}{g(s' \to s)A(s' \to s)} = \frac{p(s')}{p(s)} = \frac{e^{-\beta E_{s'}}}{e^{-\beta E_s}} = e^{-\beta(E_{s'} - E_s)} \, . \tag{1.9}$$

There is still an ambiguity in this expression for $A(s \to s')$, but recalling that T must be $0 < T < 1$, we can resolve it. If $\Delta E < 0$, then $e^{-\beta \Delta E} > 1$, so we choose $A(s' \to s) = e^{\beta \Delta E}$ so that $A(s' \to s) = 1$. In the case $\Delta E < 0$, we simply have $A(s \to s') = e^{-\beta \Delta E}$. This is neatly summarized as the Metropolis-Hasting acceptance ratio

$$A(s \to s') = \min\left(1, \frac{p(s')g(s' \to s)}{p(s)g(s \to s')}\right) \, . \tag{1.10}$$

Intuitively, it makes sense that a state will jump to a lower energy (higher probability) state with probability one, yet will jump to a higher energy state only with exponentially vanishing probability. While theoretically the exponential tail allows any state to transition to any other state, in practice, large jumps are not likely. This results in the mode-mixing problem. Even worse, if the state does jump, it is likely to transition back in the next step.

One problem with the Metropolis-Hastings algorithm is that the samples produced will not be independent as required by Eq. 1.4. In fact, since Markov chain depends only on each previous state, each successive sample is highly correlated. One typically has to compute how many Monte Carlo steps are needed for two samples become roughly uncorrelated. One measure is the autocorrelation function

$$C_{\mathcal{O}}(t) \propto \langle \mathcal{O}(i)\mathcal{O}(i+t) \rangle - \langle \mathcal{O}(i) \rangle \langle \mathcal{O}(i+t) \rangle \tag{1.11}$$

which typically goes as $C_{\mathcal{O}}(t) \sim \exp(-t/\tau_{\text{auto}})$. Choosing a transition so that samples are dissimilar typically results in lower autocorrelation times. For the Ising model, cluster updates such as the Swendson-Wang [38] or Wolff update [39] dramatically lower the autocorrelation time.

1.1.4 Gibbs sampling

In the Metropolis-Hastings algorithm, we only used ratios of the probabilities $p(s')/p(s)$ and did not assume anything about the form of p. Another common form of sampling is Gibbs sampling, which can we regarded as a subset of the Metropolis-Hastings with more structure imposed on p. In the literature, this is also sometimes known as the heat bath algorithm or Glauber dynamics. Gibbs sampling is also common in Bayesian inference, and plays a crucial role in the training of restricted Boltzmann machines [40].

In Gibbs sampling, all but one random variable is fixed, so that we can sample from a *univariate* conditional distribution instead of the full high-dimensional distribution. Sampling from a univariate distribution is much easier since the dimension of the space is greatly reduced. The benefit of Gibbs sampling is when the conditions $p(x_i|\boldsymbol{x}_{\neq i})$ can be computed exactly using Bayes' rule. To satisfy ergodicity, each conditional must be non-zero.

Gibbs sampling is a local update algorithm. We rely on updating spin s_i with s_i' according to the transition probability $g(s_i \to s_i')$. Although it might seem unfamiliar, Gibbs sampling is a subset of Metropolis-Hastings. Since Gibbs involves an update only on site i, the proposal probabilities are

$$g(\boldsymbol{s} \to \boldsymbol{s}') = p(s_i'|\boldsymbol{s}_{\neq i})p(\boldsymbol{s}_{\neq i}) \tag{1.12}$$
$$g(\boldsymbol{s}' \to \boldsymbol{s}) = p(s_i|\boldsymbol{s}_{\neq i})p(\boldsymbol{s}_{\neq i}) \tag{1.13}$$

Notice that $p(\boldsymbol{s}_{\neq i})$ is the same for both transitions since only a local change on site i is considered. The Metropolis-Hasting ratio becomes

$$\frac{p(\boldsymbol{s}')g(\boldsymbol{s}' \to \boldsymbol{s})}{p(\boldsymbol{s})g(\boldsymbol{s} \to \boldsymbol{s}')} = \frac{p(\boldsymbol{s}')}{p(\boldsymbol{s})}\,\frac{p(s_i|\boldsymbol{s}_{\neq i})p(\boldsymbol{s}_{\neq i})}{p(s_i'|\boldsymbol{s}_{\neq i})p(\boldsymbol{s}_{\neq i})} \tag{1.14}$$
$$= \frac{p(\boldsymbol{s}')}{p(\boldsymbol{s})}\,\frac{p(\boldsymbol{s})}{p(\boldsymbol{s}')} \tag{1.15}$$
$$= 1\,. \tag{1.16}$$

We see that the Gibbs step is always accepted. This is called a rejection-free algorithm.

Since Gibbs sampling spin s_i only depends on the nearest neighbours, many spins can be updated in parallel. For instance, in a one-dimensional lattice, we marginalize out all even (or odd) spins with

$$p(\boldsymbol{s}_{\text{even}}|\boldsymbol{s}_{\text{odd}}) = \prod_{\text{odd}\, i} p(s_i|\boldsymbol{s}_{\text{odd}})\,, \tag{1.17}$$
$$p(\boldsymbol{s}_{\text{odd}}|\boldsymbol{s}_{\text{even}}) = \prod_{\text{even}\, j} p(s_j|\boldsymbol{s}_{\text{even}})\,. \tag{1.18}$$

The method of alternating between two or more independent variables is referred to as block Gibbs sampling. In this method, sampling is very fast since there are no rejections and many spins can be updated simultaneously. This is important for the restricted Boltzmann machine in Chapter 4. However, as a local update algorithm, Gibbs sampling still suffers from poor ergodicity, i.e., the mode-mixing problem.

Consider the example of the classical Ising model in d-dimensions which has the Hamiltonian

$$H = -J \sum_{\langle ij \rangle} s_i s_j - h \sum_i s_i \tag{1.19}$$

where $\sum_{\langle ij \rangle}$ denotes a sum over nearest neighbour spins. A sample $\mathbf{s} = (s_1, s_2 \ldots s_N)$ is a configuration of N spins with energy $E(\mathbf{s})$. The statistics follow the Boltzmann distribution from Eq. 1.1. Let $\boldsymbol{s}_{\neq i} = (s_1, s_2, \ldots s_{i-1}, s_{i+1} \ldots s_N)$ be the same configuration without the ith spin s_i. It is clear that the only difference in the energy of the states will be from the ith spin term:

$$\begin{aligned} E(\boldsymbol{s}) &= E(\boldsymbol{s}_{\neq i}) + E(s_i) \\ &= E(\boldsymbol{s}_{\neq i}) - h s_i - J \sum_{\langle ij \rangle} s_i s_j \, . \end{aligned} \tag{1.20}$$

Using Bayes' rule, the probability that spin $s_i = 1$, with all others fixed is

$$p(s_i | \boldsymbol{s}_{\neq i}) = \frac{p(\boldsymbol{s})}{p\left(\boldsymbol{s}_{\neq i}\right)} \tag{1.21}$$

$$= \left(\frac{e^{-\beta E(\mathbf{s})}}{\sum_{\{\mathbf{s}\}} e^{-\beta E(\mathbf{s})}} \right) \left(\frac{\sum_{\{\mathbf{s}_{\neq i}\}} e^{-\beta E(\mathbf{s}_{\neq i})}}{e^{-\beta E(\mathbf{s}_{\neq i})}} \right) \tag{1.22}$$

$$= e^{-\beta E(\mathbf{s}) + \beta E(\mathbf{s}_{\neq i})} \left(\frac{\sum_{\{\mathbf{s}_{\neq i}\}} e^{-\beta E(\mathbf{s}_{\neq i})}}{\sum_{\{\mathbf{s}\}} e^{-\beta E(\mathbf{s})}} \right) \tag{1.23}$$

The normalization constants simplify by writing out the sums over each s_i spin:

$$p(s_i|\boldsymbol{s}_{\neq i}) = e^{-\beta E(s_i)} \left(\frac{\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta E(\mathbf{s}_{\neq i})}}{\sum_{s_i=\pm 1} \left(\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta E(\mathbf{s}_{\neq i}) + \beta E(s_i)} \right)} \right) \tag{1.24}$$

$$= e^{-\beta E(s_i)} \left(\frac{\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta E(\mathbf{s}_{\neq i})}}{\sum_{s_i=\pm 1} e^{\beta E(s_i)} \left(\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta E(\mathbf{s}_{\neq i})} \right)} \right) \tag{1.25}$$

$$= \frac{e^{-\beta E(s_i)}}{\sum_{s_i=\pm 1} e^{\beta E(s_i)}} \tag{1.26}$$

$$= \frac{e^{\beta h + \beta J \sum_{\langle j \rangle_i} s_j}}{2 \cosh\left(\beta h + \beta J \sum_{\langle j \rangle_i} s_j \right)} \tag{1.27}$$

$$= \frac{1}{1 + e^{-\beta (h + J \sum_{\langle j \rangle_i} s_j)}} \tag{1.28}$$

where we used the fact that the sum over $s_i = \pm 1$ is symmetric. Due to nearest neighbours interactions, only a few spins (the neighbours) directly affect spin s_i. Sampling proceeds by iterating through all lattice sites $i = 1, \ldots, N$ and setting $s_i = 1$ with probability $p(s_i|\boldsymbol{s}_{\neq i})$.

1.1.5 Autoregressive sampling

So far, the methods discussed rely on a Markov chain to produce samples which suffer from correlations. While this scales well to high-dimensional problems, in some cases we can improve sampling significantly by producing truly independent samples. In this Section, we describe autoregressive sampling: a method of generating samples from a distribution of many variables from knowledge of all conditional distributions. In condensed matter, this is called *perfect sampling* in the context of tensor networks [41].

For a vector $\mathbf{x} = (x_1, x_2, \ldots x_N)$, the chain rule for probabilities states that the total probability of $\mathbf{x}$ is the product of all conditional probabilities

$$p(\mathbf{x}) = \prod_{i=1}^{N} p\left(x_i | \mathbf{x}_{<i}\right) \tag{1.29}$$

$$= p(x_1)\, p(x_2|x_1)\, p(x_3|x_1, x_2) \ldots p(x_N|x_1 \ldots x_{N-1}), \tag{1.30}$$

where $\mathbf{x}_{<i} = (x_1, x_2, \ldots, x_{i-1})$ denotes all elements with index less than i. This is called the autoregressive property because of origins in time-series models. However, instead of times-series, we can enumerate sites on a lattice or other quantum numbers.

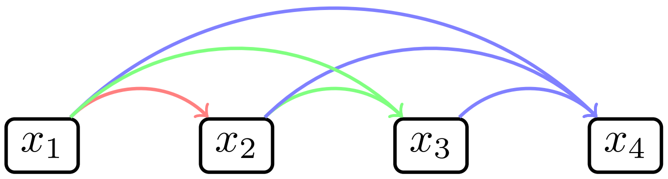


Figure 1.1: Product rule of probabilities as a graphical model for an autoregressive Bayesian network with four nodes.

For instance, consider the state $\psi \propto |010\rangle + |100\rangle + |111\rangle$. We first learn that the probability of the first qubit to be in $|0\rangle_1$ is $p(0_1) = \frac{1}{3}$. The probability of the next qubit being in either $|0\rangle_2$ or $|1\rangle_2$ is given by the four conditionals

$$p(0_2|0_1) = 1 \tag{1.31}$$

$$p(0_2|1_1) = 0 \tag{1.32}$$

$$p(1_2|0_1) = \frac{1}{2} \tag{1.33}$$

$$p(1_2|1_1) = \frac{1}{2} \tag{1.34}$$

Likewise for the third qubit. This example makes it clear that any errors in the predictions of earlier probabilities (i.e., $p(x_1), p(x_2|x_1)\ldots$) propagate to the last qubits (i.e. $p(x_N|\mathbf{x}_i < N)$). Notice that the complexity of this representation grows exponential in the length of the sequence N. Analogous to Monte Carlo where we cannot enumerate all the configurations, in autoregressive models we cannot enumerate all the conditional probabilities.

We simplify the description by recursively defining the conditions to depend on previous conditionals

$$p\left(s_i|\boldsymbol{s}_{<i}\right) = F\left(p\left(s_{i-1}|\boldsymbol{s}_{<i-1}\right)\right) \, . \tag{1.35}$$

This is the basis of recurrent neural networks and deep autoregressive models.

Autoregressive sampling for the one-dimensional Ising model is trivial since each conditional reduces to the Gibbs result from Eq. 1.28, i.e.,

$$p(s_i|\boldsymbol{s}_{<i}) = p(s_i|s_{i-1}) = \frac{1}{1 + e^{-\beta(h + J\sum_{\langle j\rangle_i} s_j)}} \tag{1.36}$$

In essence, the nearest neighbour interactions are too simple for autoregressive models to provide a benefit.

The simple result for the Ising model is not indicative of the ability of this method. For instance, natural language possess much stronger dependence on proceeding words and hence contains longer-range correlations. In language modelling, Markov models have mostly failed, whereas autoregressive model have achieved groundbreaking results [42, 43].

1.1.6 Summary

In many ways, Monte Carlo has overwhelmingly been the greatest general purpose tool for tackling difficult and high-dimensional problems. For condensed matter physics, these are essential to probe the nature of entanglement in quantum states [44, 45] and many other interacting many-body phenomena.

In this Section, we discussed the essential properties of stochastic sampling techniques that are used throughout this Thesis. We introduced Metropolis-Hasting, a standard technique for many areas of physics. Further, we showed that Gibbs sampling is a rejection-free algorithm and is a special case of the Metropolis-Hastings algorithm that is used in machine learning graphical models. We also briefly described autoregressive sampling, an approach that is very effective in modern machine learning.

In the next Section, we introduce machine learning methods: a data-driven approach that has profoundly changed the research landscape in computer science. These promising methods may prove to have the widespread success that Monte Carlo experienced in the past 50 years.

1.2 Machine Learning

In an influential blog post, Karpathy (Director of AI at Tesla) argues that machine learning is a fundamental new paradigm for computing: it represents Software 2.0 [46]. In the old days of Software 1.0, people would design algorithms to accomplish a specific task. The language they used was computer code such as C, Python, Java, etc., which compiles down to machine instructions. This method has powered many technologies such as the internet, autopilot in airplanes, risk assessment in financial markets, simulations of planet formation and statistical physics. On the other hand, machine learning proposes that programs be written in a more abstract way. The language is the unfamiliar linear algebra manipulations of the parameters of a neural network. Typical networks can involve millions, even billions of parameters[5] that humans are simply not capable of programming manually.

Machine learning is commonly broken down into three main areas: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, we task the algorithm with optimally matching a label to an input. The algorithm is repeatedly shown instances of inputs and labels, with the hope that it can detect what makes a particular input have a certain label. Most of the success of machine learning has been in supervised learning.

In unsupervised learning, we are tasked with finding structures with only the input data. Such examples include clustering algorithms that partition the data into different regions based on a learned similarity metric. Another unsupervised task is density estimation. From observed data, we try to learn the underlying probability density that produced that data. The goal is to not only reproduce training samples, but also generate new ones not in the training set.

Reinforcement learning is somewhat distinct from the other two areas because it does not involve a training data set. Instead, we model an agent interacting with its environment through a series of rewards to encourage good performance. The agent wants to maximize the sum of future rewards. There is a trade-off between exploring new actions to find better rewards, or saying with a safe, but small reward. Reinforcement learning has been the source of breakthrough in games such as StarCraft II [49] and Go [50].

In the majority of this Thesis, we focus on supervised learning and generative modelling. For that purpose, we introduce supervised learning in the following Sections, and only generative models as needed in subsequent Chapters. Starting with polynomial regression, we cover neural networks and stochastic gradient descent before concluding with an example of classifying phases in the Ising model.

1.2.1 Supervised learning: polynomial regression

Supervised learning deals with a task of correctly labelling of input x with a label y. It is called supervised because we provide *training* data consisting of pairs (x, y). A machine learning model is a function that predicts the label given the input, $f(x)$. When the labels are discrete, this problem is called *classification*, while for continuous labels it is called *regression*. But how do we choose the function f?

[5]For instance, recent works have used 8 and 175 billion respectively [47, 48].

In the case of regression, we could use a polynomial fit,

$$f_\theta(x) = \theta_n x^n + \theta_{n-1} x^{n-1} + \ldots + \theta_0 \tag{1.37}$$

as an approximation. Alternatively, we can specify a vector of inputs $\boldsymbol{x} = (x_1, \ldots x_n)$ and use a linear approximation

$$f_\theta(\boldsymbol{x}) = \sum_i \theta_i x_i + \theta_0 \,. \tag{1.38}$$

If we choose each $x_i = x^i$, we recover the polynomial fit. This is called *feature engineering.* While the polynomial case is a trivial example, it will be important for neural networks.

How do we quantify the best fit? The likelihood function measures the probability of observing y given the data points x. If the all data points are independent, then the likelihood factorizes to the product of each individual likelihood $L(y, x) = \prod_i L(y_i, x_i)$. If we assume that the random variable y_i follows the true model $f_\theta(x_i)$ up to Gaussian noise[6] ϵ, then each $L(y_i, x_i)$ is a normal distribution. Hence, for the total data set y and x, the likelihood is

$$L(y, x) \propto \prod_i \exp\left[-\frac{(f_\theta(x_i) - y_i)^2}{2\sigma^2}\right] . \tag{1.39}$$

To find the best fit, we simply maximize the likelihood. It is more convenient to work with the log-likelihood instead,

$$\log L(y, x) \propto -\frac{1}{2\sigma^2} \sum_i (f_\theta(x_i) - y_i)^2 + \text{const} \,. \tag{1.40}$$

Since this is monotonic, it does not change the optimization. We see that maximizing the log-likelihood is equivalent to minimizing the squared error loss

$$\mathcal{L}(\theta) = \sum_i \left(f_\theta(x_i) - y_i\right)^2 \,. \tag{1.41}$$

To find the best fit, we simply minimize the difference between the prediction and true value. We denote the optimal values of the parameters as θ^*.

How do we optimize the parameters θ? The most straightforward method is to use gradient descent. The parameters are updated according to the gradient of the loss function

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L} \tag{1.42}$$

[6]Compactly written as $y_i \sim f_\theta(x_i) + \epsilon_i$ where $\epsilon \sim \mathcal{N}(0, \sigma)$.

with a step size η. With a small enough step size, the optimization is guaranteed to converge to a local minimum. For the least squares loss, we have quadratic dependence on the parameters so the gradient is linear in θ,

$$\nabla\mathcal{L}(\theta) = 2\sum_i (f_\theta(x_i) - y_i) \ . \tag{1.43}$$

So far we have two *hyperparameters*: the degree of the polynomial fit n, and the gradient descent step size η. A hyperparameter is not adjustable within the framework of the optimization. However, clearly changing the degree n has a dramatic impact on the fit.

Consider a training data set with $y_i = \sin(x_i)$ up to some random noise. In Fig. 1.2, we show three different fits for different degrees n. The first fit, $n = 2$ is clearly *underfitting* since it does not seem to capture the pattern in the data. The second choice $n = 3$ is in reasonably good agreement. Lastly, $n = 10$ manages to directly intersect every single data point in the training set. But is this a good fit?

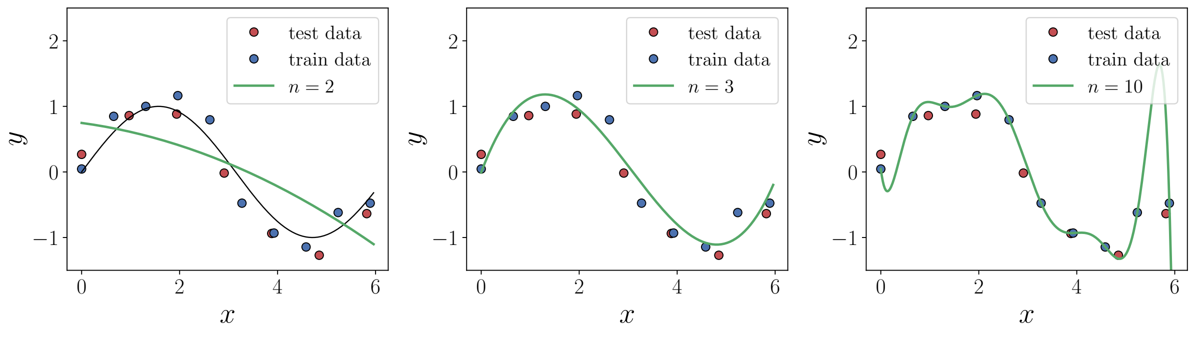


Figure 1.2: Noisy data for the function $y = \sin(x)$. We compare three different polynomial fits of degree $n = 2, 3, 10$. The training data is in blue. The red points are the test set which is *not* used to train the model.

We can consider testing the generalization of fit to new data points. This *test set* is shown in the red points in Fig. 1.2. We can judge a fit based on the loss function evaluated on the optimal parameters θ^* on both the training and test sets. Even though for $n = 10$, we perfectly fit the training data, the loss is much worse on the test set. This is called *overfitting* and is very difficult to avoid in general. According to the test errors, the best fit is for $n = 3$ which confirms our intuition from the plots in Fig. 1.2. Generally, we want a model which minimizes the generalization gap: the difference between the test and training error. Figure 1.3 shows the expected generalization gap as increasing the degree n.

To reduce the chance of overfitting, we need a way of favouring simple models. One way is to introduce a *regularization* term into the loss function. The goal is to encourage the model to use small parameters. A common choice is L2 regularization which adds the square of each parameter to the loss function with a hyperparameter λ that determines the strength of the regularization. The effective loss function is

$$\tilde{\mathcal{L}} = \sum_i \left(f_\theta(x_i) - y_i\right)^2 + \lambda \sum_i \theta_i^2 \,. \tag{1.44}$$

In considering polynomial regression, we have covered all essential topics for understanding neural networks. We first introduced feature engineering, which could be polynomial or more exotic functions of the input. We discussed the importance of hyperparameters, both for representation ability and for optimization. Further, we used gradient descent to minimize a loss function such as least squares. We found that overly complex models can easily overfit the data, leading to poor generalization. To combat overfitting, we used regularization of the loss function to favour simple functions. All of these concepts are directly applicable to neural networks that we use throughout this Thesis.

1.2.2 Neural networks

Inspired by biological neurons in the brain, artificial neural networks offer a powerful method for classification problems. A real neuron interacts via dendrites receiving electrical

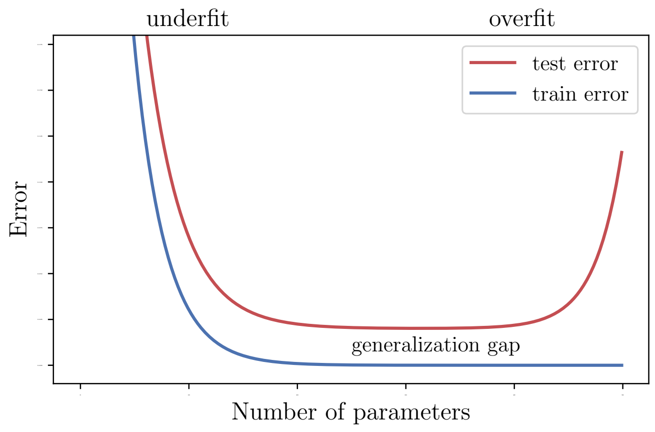


Figure 1.3: Overfitting occurs when the test error increases while the training error decreases. The difference between training and test error is called the generalization gap.

signals from the axons of other neurons. In a *perceptron*, this is represented by a function $f_\theta(\boldsymbol{x}) = \Theta(\sum_i W_i x_i + b)$ where Θ is the step function and W, b the weights and biases respectively. This is a linear classifier that predicts $\hat{y} = 1$ when $\sum_i W_i x_i + b > 0$ and zero otherwise.

The theory of perceptrons was developed in Rosenblatt [51] in the 1950s. Perceptrons are a universal form of computation since they can represent simple logical gates such as the NAND gate. Despite this, perceptrons are difficult to train due to an inefficient training algorithm. The key difficulty is the lack of a differentiable activation function of the neuron that prohibits gradient based optimization.

In modern machine learning, we typically choose a differentiable activation function. Replacing the step-function in the perceptron with a smooth sigmoid function results in neural networks that predicts an output probability opposed to a definite class label. This is much more useful since we can see that a network is unsure about a certain input if it assigns a 50% probability of either class. The sigmoid function shown in Fig. 1.4 is a common choice. Note the derivative of the sigmoid function is $\nabla\sigma = \sigma(1-\sigma)$.

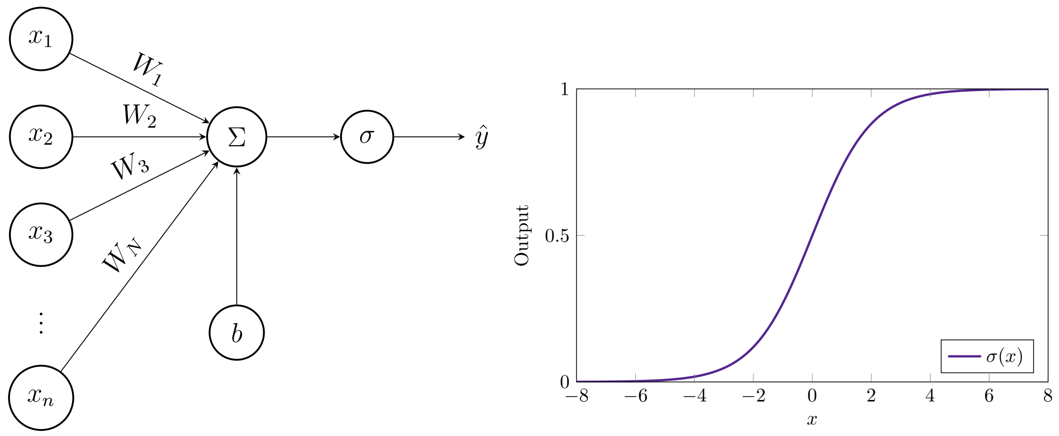


Figure 1.4: (left) Single neuron with sigmoid activation function. The inputs $x_1, \ldots, x_n$ are multiplied by the weights $W_1, \ldots, W_n$ and added to b before a sigmoid activation function is applied. The output is the class prediction $\hat{y}$. (right) Sigmoid activation function. As $x \to \infty$, the output saturates to $\sigma(x) = 1$, and conversely $\lim_{x\to-\infty} \sigma(x) = 0$.

A full sigmoid neuron is depicted in Fig. 1.4. It consists of a linear function with adjustable parameters, followed by a non-linear sigmoid activation function.

$$f_\theta(\boldsymbol{x}) = \sigma\left(\sum_i W_i x_i + b\right) \tag{1.45}$$

This is a smooth version of the perceptron linear classifier. By itself, a single neuron cannot separate complex classes, but by stacking many neurons together, it can classify non-linear features.

A multi-layer neural network is a composition of many linear and non-linear transformations as shown in Fig. 1.5. Consider a network with L layers where the width of the ℓth layer is n_ℓ. The vector of hidden units $\boldsymbol{h}^\ell$ is

$$\boldsymbol{h}^\ell = \sigma\left(\boldsymbol{W}^\ell \boldsymbol{h}^{\ell-1} + \boldsymbol{b}^\ell\right) \tag{1.46}$$

where $\boldsymbol{W}$ is the $n_\ell \times n_{\ell-1}$ weight matrix, and $\boldsymbol{b}^\ell$ the vector of biases with size n_ℓ.

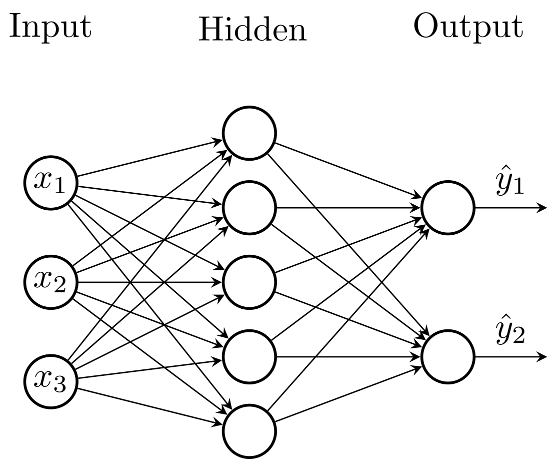


Figure 1.5: A multi-layer neural network. Inputs x_i are fed into a hidden layer before outputting class predictions $\hat{y}_i$.

Multi-layer networks have strong representational power according to the *universal approximation theorem* [52, 53, 54] meaning they can approximate any function given enough neurons. Although this is an important result, it does not quantify how many neurons are needed for a given problem. Moreover, it fails to mention two other important factors for practical learning: learnability and efficiency. The sigmoid neural network has greater learnability since it allows optimization by using gradients.

There are three ways to compute gradients: exact expressions, numerically with finite-difference, and automatic differentiation. Exact expressions offer the fastest and most accurate gradients, however they are often infeasible for complex models. We use exact expressions in Chapter 4 when training a restricted Boltzmann machine and in Chapter 5 for optimizing a variational model. Finite-difference schemes are common in scientific computing, but can be numerically unstable and scale poorly to large dimensions. Automatic differentiation (AD), in contrast, involves the accumulation of gradient from each step in program. In essence, we use the chain rule to decompose a large gradient into many smaller steps. Going through each step we store the exact gradient of that step and combine it with the previous one, thus building the complete gradient.

AD typically comes at the expense of extra memory for storing values while traversing a program. However, AD provides a gradient up to machine error with no approximations. A good introduction to AD is the textbook by Griewank and Walther [55], or the review by Baydin and Pearlmutter [56] that focuses on neural network applications.

To minimize the loss function, we need the gradient of the loss $\nabla_\theta \mathcal{L}$. AD is essentially using the chain rule for differentiation to break down a complex expression into smaller parts. Consider the single activation function from Fig. 1.4 and only one data point so the loss is $\mathcal{L} = (\hat{y} - y)^2$. The first term is $\frac{\partial \mathcal{L}}{\partial \hat{y}}$. For the least square loss function from Eq. 1.41, that is simply $2 \sum_j (y_j - \hat{y}_j)$. The second term is the gradient from the sigmoid activation function, one can easily check that $\nabla\sigma(x) = \sigma(x)(1-\sigma(x))$. Lastly, we have the term $\frac{\partial A}{\partial W_i}$ where we use $A = \sum_i W_i j x_i + b$ for brevity. Together this yields

$$\frac{\partial \mathcal{L}}{\partial W_i} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W_i} \tag{1.47}$$

$$= \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \sigma} \frac{\partial \sigma(A)}{\partial A} \frac{\partial A}{\partial W_i} \tag{1.48}$$

$$= 2\,(y - \hat{y})\,(\hat{y}(1-\hat{y}))\,(x_i) \tag{1.49}$$

$$= 2\,x_i\,\hat{y}\,(1-\hat{y})\;(y-\hat{y})\;. \tag{1.50}$$

In a multi-layer network this quickly becomes difficult to compute by hand. Fortunately, each step in the chain rule contains only a simple derivative. The process of AD records the computations of the network in a graph and recursively iterates through each node while adding its derivative.

1.2.3 Stochastic gradient descent

We update the parameters via the gradient step

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L} \tag{1.51}$$

where η is the step size, also called the learning rate. Choosing the best learning rate can be difficult. Too small of a learning rate leads to slow convergence, while too large a rate will never converge. Some gradient-based algorithms attempt to change the learning rate adaptively during training.

When using a large training set, it may become expensive to compute the gradient for the entire training data. Instead, we use a stochastic approximation. In stochastic gradient descent, we draw a sample from the training set at random, and compute the gradient for that sample. Repeating this many times gives an average over the whole training set.

Instead of single samples, it is more common to use a batch of samples. One common choice is a batch size of 32 samples. The smaller the batch size, the more stochastic the trajectory becomes as shown in Fig. 1.6. The high-variance in the stochastic gradient is both a blessing and a curse. It can help escape sharp minima, but it also may not be as direct in finding the global minima. It is an open question as to why stochastic gradient descent is so effective for deep learning [57].

In high-dimensional optimization landscapes, naive gradient descent sometimes oscillates around "canyons" where one parameter has little effect while another parameter has a drastic effect. One method for avoiding these traps is adding *momentum* to the gradient. Instead of relying only on the current gradient, we take the weighted sum with the previous gradient. We write this as

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_t - \gamma \nabla_\theta \mathcal{L}_{t-1} \tag{1.52}$$

With too high a momentum, the optimization could skip important minima; with too little, the gradient many oscillate indefinitely.

Adaptive methods change the learning rate during the optimization. For instance, Adagrad [58] uses a step that is different for each parameter θ_i by modifying it by the sum of all previous gradients. Another popular method is the adaptive moment estimation (Adam) optimizer [59]. Adam keeps the average of past squared gradients, along with an exponentially decaying average of past gradients. In this way it retains a full memory of past squared gradients, but also emphasizes that more recent gradients are more important. A comparison of these algorithms is shown in Fig. 1.6.

We now have all the tools for understanding neural networks. In the next Section, we consider an example of classifying phases from an Ising model where we put together all the steps so far: network architecture, regularization, gradient optimization through AD, and measuring the generalization gap.

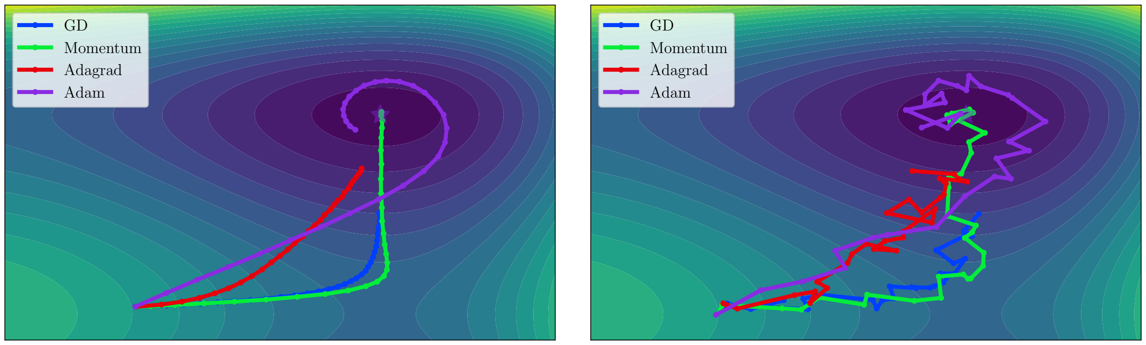


Figure 1.6: (left) Gradient based optimizations with a large batch size has a smoother trajectory. (right) A smaller batch size introduces more noise into the gradient trajectory. For both plots, gradient descent is compared with the momentum method, Adagrad, and Adam optimizers.

1.2.4 Classifying phases of the Ising model

As an example of a classification problem for physics, consider the two-dimensional Ising model on a square $L \times L$ lattice. This model was considered by Carrasquilla and Melko as one of the first uses of machine learning in condensed matter physics [60]. There is a critical point at $T_c \approx 2.269$ that separates the low-temperature ferromagnetic phase from the high-temperature paramagnetic phase. In the ferromagnet, spins favour alignment, while in the paramagnet, they are suspect to thermal fluctuations. The phases can be distinguished classically with an order parameter, in this case the magnetization shown in Fig. 1.7. Standard Monte Carlo from Chapter 1.1 is used to generate snapshots of configurations of the system. Typically, we could estimate the magnetization to distinguish the phases. However, we will assume no knowledge about the underlying physics, instead we only rely on the ability of a neural network to distinguish the phases.

We train a multilayer sigmoid network for binary classification of phases. The input is an $L \times L$ vector of binary Ising spins, which feed into a 100-neuron hidden layer with sigmoid activation functions. The final output is a single number $\hat{y}$ that gives the probability of either the paramagnetic or ferromagnetic phase. Instead of the least square loss, we use the common cross-entropy loss function $\mathcal{L} = \sum_i y_i \log \hat{y}_i$[7]. We train the neural network with stochastic (batch) gradient descent on a training set of 10,000 samples at 32 different

[7]The least squares function suffers from saturating gradients in Eq. 1.47, whereas the cross entropy loss does not. Also, cross entropy loss for binary classification motivates the interpretation of the output as a probability.

temperatures. To mitigate overfitting, we add L2 regularization to the loss function. We further vary the system size from $L = 16$ to $L = 32$ in increments of four.

The network readily achieves small training error, but the important question is testing error. In Fig. 1.7, the probability of the ferromagnetic phase is shown in blue. We find the probability is near unity for low-temperatures, yet reaches 0.5 at the critical temperature. This is the point of maximal uncertainty of the network.

Another property is that the transition becomes sharper as system size increase. This is consistent with the thermodynamic behaviour and finite-size samples from Monte Carlo in Fig. 1.7. Carrasquilla and Melko [60] go on to estimate T_c using the point of maximal uncertainty using the network predictions and find consistent finite-size scaling.

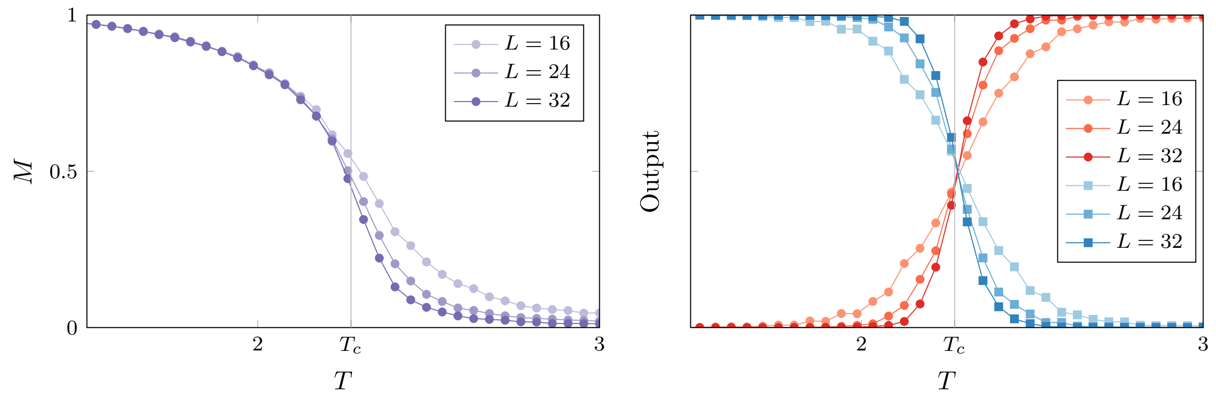


Figure 1.7: (left) Magnetization M for the two-dimensional classical Ising model. Sampling was performed with the Metropolis-Hasting algorithm. (b) Output probability for binary classification of phases with a neural network.

Why does the network correctly classify the phases? Having seen many training examples, the network has extracted features that distinguish the phases. A simple feature is the magnetization, $M = \frac{1}{N}\sum_i^N s_i$. If the input to the final neuron is $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, we need only that $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \propto M$ to classify based on a threshold magnetization. This is indeed what the authors find in [60].

Generally, neural networks can encode much more complex features such as facial shapes, textures, or spectral filters [16]. The Ising problem serves as a toy model for classification where we gain some understanding as to how the network does its job.

1.2.5 Summary

Machine learning is a powerful black box method for the study of interacting systems. Analogous to image recognition, neural networks can perform classification of physical configurations by learning statistical features. Despite widespread success, there are many open questions as to why neural networks are so effective in supervised learning. By using machine learning on well understood physical models, we can gain insight into their effectiveness and generalization ability.

Aside from pure machine learning approaches, automatic differentiation is a general framework that may impact scientific computing. Differentiable programming languages will allow code to do an introspection of its own code to computer gradients with respect to any parameters. Many commonly used numerical routines can be differentiated through, including linear algebra operations, singular-value decompositions, fluid-dynamics calculations, and solving ordinary differential equations. The flexibility provided by differentiable programming will aid in the development of advanced and robust new algorithms. Various examples already include optimizing tensor networks [61], quantum circuit simulators [62], and quantum chemistry [63].

In the remainder of this Thesis, we will frequently use the terminology and ideas from this Section.

2

Machine Learning Topological Defects

In this Chapter, we investigate whether neural networks can learn to classify phases based on topological defects. We address this question on the two-dimensional classical XY model which exhibits a Kosterlitz-Thouless transition. We find significant feature engineering of the raw spin states is required to convincingly claim that features of the vortex configurations are responsible for learning the transition temperature. Further, we show that a single-layer network does not correctly classify the phases of the XY model, while a convolutional network easily performs classification by learning the global magnetization. Finally, we design a deep network capable of learning vortices without feature engineering. We demonstrate the detection of vortices does not necessarily result in the best classification accuracy, especially for lattices of less than approximately 1000 spins. For larger systems, it remains a difficult task to learn vortices.

This Chapter consists of work previously published in Physical Review B [1] and an accompanying blog post written for the website `physicsml.github.io` [2].

2.1 Introduction

Motivated by the successful application of supervised learning to conventional symmetry breaking transitions [60], it is natural to ask whether neural networks are capable of detecting unconventional phase transitions driven by the emergence of topological defects. The prototypical example for such a system is the two-dimensional XY model, which exhibits a Kosterlitz-Thouless (KT) transition [64, 65, 66]. Transitions of this universality class can be found in a variety of systems, with one of the most famous being the superfluid transition in two-dimensional helium [67, 68, 69].

Several unsupervised learning strategies have been applied to this model, for example, it was found that principal component analysis (PCA) performed on spin configurations effectively measure the magnetization [70, 71, 72]. Even when trained directly on vorticity, PCA is unable to resolve vortex-antivortex unbinding, which is attributed to the linearity of this method [71]. Similarly, variational autoencoders [73], a popular tool for unsupervised learning based on Bayesian inference, perform classification by learning a bulk magnetization [74, 70, 72].

In contrast, efforts in supervised learning have been more successful, although none have been applied directly to the XY model. Broecker et al. [75] showed that a convolutional network trained on winding numbers correctly classified interacting boson phases separated by a KT transition. However, this same method failed when trained on raw configurations. A related problem was explored by Zhang et al. [76], where the authors trained a convolutional network on Hamiltonians of one-dimensional topological band insulators labelled by their global winding number. By inspecting the trained weights, the authors deduced that the network had learned to calculate the winding number correctly.

In this Chapter, we apply several supervised machine learning strategies to identify the KT transition in the two-dimensional XY model. We ask whether it is possible for a neural network, trained only on raw spin states labelled by their phases, to learn a representation that can be interpreted as the local vorticity. In Section 2.3, we compare supervised learning algorithms involving feed-forward and convolutional neural networks applied to both unprocessed (raw spin configurations) as well as processed input data (vorticity). We then use both types of input data in the semi-supervised confusion scheme from van Nieuwenburg et al. [77] in Section 2.4. In Section 2.5, we train a supervised CNN to compute the vorticity of a sample. Lastly, in Section 2.6, we explore to which degree feature engineering of the raw spin configurations is required, and whether the network can learn to process the data into something resembling vortices using additional convolutional layers.

2.2 The Kosterlitz-Thouless transition

The simplest physical model with vortices is the two-dimensional classical XY model which consists of unit spins on a lattice and interact only with their nearest neighbours. At low

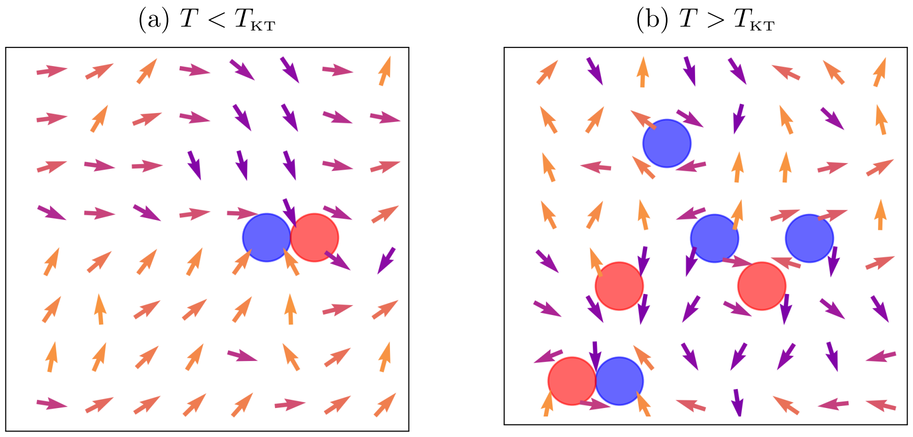


Figure 2.1: (a): A configuration of the spins in the XY model for a temperature below the KT temperature. Notice that it contains one vortex-antivortex pair that is bound together. (b): A configuration above the KT temperature contains one bound pair but also some free vortices.

temperatures, the spins generally align. However, because the spins take on continuous values, spin-wave excitations become very strong and prevent true long-range order [78]. Unlike the Ising ferromagnet, in this regime, the correlations between spins decay polynomially with their separation. This is called quasi-long-range order.

At the temperature $T_{\rm KT} = 0.8935$, the XY model exhibits a Kosterlitz-Thouless transition [79]. In contrast with conventional phase transitions, the KT transition displays no discontinuity in any observable such as the magnetization or energy. In this sense, it is considered an infinite-order transition. The transition is caused by the unbinding of vortex-antivortex pairs above $T_{\rm KT}$. Below this temperature, it takes infinite energy to excite a single vortex; however, thermal fluctuations can create vortex-antivortex pairs so long as they remain bound together (Figure 2.1). Above $T_{\rm KT}$, it is entropically favourable for vortices to separate. This balancing act between energy and entropy is responsible for the vortex-unbinding KT transition.

The classical XY model is described by the Hamiltonian

$$\mathcal{H}_{XY} = -J \sum_{\langle ij \rangle} \cos\left(\theta_i - \theta_j\right), \tag{2.1}$$

where $\langle ij \rangle$ indicates that the sum is taken over nearest neighbours and the angle $\theta_i \in [0, 2\pi)$ denotes the spin orientation on site i. The topological defects in the XY model are quantified through the vorticity

$$\oint_C \nabla\theta \cdot d\vec{\ell} = 2\pi w, \qquad w = \pm 1, \pm 2, \dots, \tag{2.2}$$

where C denotes any closed path around the vortex core and w is the winding number of the associated spins. A vortex is defined by positive winding number, $w = 1$, and an antivortex by $w = -1$. On a lattice, the integral may be approximated by the sum of the angle differences over a plaquette. An example of a vortex and antivortex is shown in Fig. 2.2.

The essential singularity of the free energy at T_{KT} means that all derivatives are finite at the transition. However, the specific heat is observed to be smooth at the transition, with a non-universal peak at a $T > T_{\text{KT}}$ which is associated with the entropy released when most vortex pairs unbind [80]. While the thermodynamic limit of the XY model has strictly zero magnetization for all $T > 0$, a non-zero value is found for systems of finite size (see Fig. 2.3b) [79, 81]. This in particular makes learning the topological features of the model difficult for machine learning algorithms.

One method to calculate T_{KT} from finite-size data is to exploit the Nelson-Kosterlitz universal jump [82, 83, 79]. This is determined from where the helicity modulus, Υ, crosses $\frac{2T}{\pi}$. The helicity modulus, also called spin wave stiffness or spin rigidity, measures the response of a system to a twist in the boundary conditions (i.e., torsion). From the linearized renormalization group (RG) equations, one can derive the finite-size scaling behaviour of the critical temperature $\tilde{T}_{\text{KT}}$ on a $L \times L$ lattice to be

$$\tilde{T}_{\text{KT}}(L) \approx T_{\text{KT}} + \frac{\pi^2}{4c(\log L)^2}\,, \tag{2.3}$$

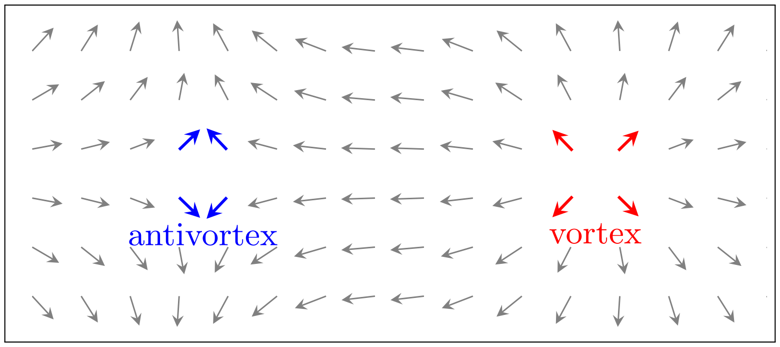


Figure 2.2: An example of a vortex and antivortex in the XY model on the lattice. A vortex has winding number $w = 1$, while an antivortex has $w = -1$.

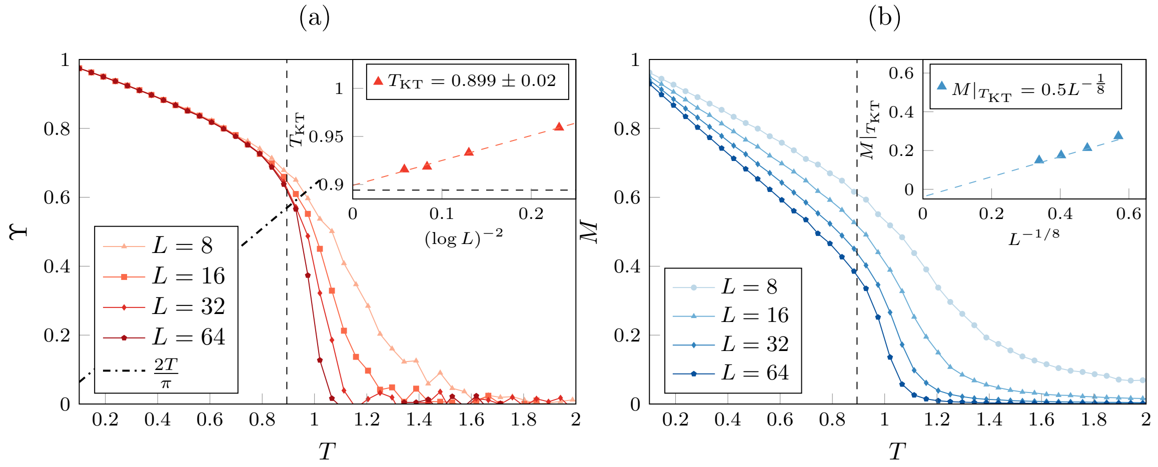


Figure 2.3: Estimators of the XY model on a $L \times L$ lattice with periodic boundary conditions computed via Monte Carlo sampling. (a) The helicity modulus for various lattice sizes L. The estimated critical point $\tilde{T}_{\rm KT}$ is determined by the Nelson-Kosterlitz universal jump where the helicity modulus, Υ, intersects the line $\frac{2T}{\pi}$. The inset shows how $\tilde{T}_{\rm KT}$ scales with $(\log L)^{-2}$ towards the thermodynamic $T_{\rm KT}$ shown by the black dashed line. (b) The non-zero magnetization present in the finite-size XY model. The magnetization vanishes as $L^{-\frac{1}{8}}$ in the thermodynamic limit with the scaling shown in the inset.

with a constant c [82]. Fig. 2.3a shows the helicity modulus Υ and the scaling of $T_{\rm KT}$ derived from Monte Carlo simulations.

We employ standard Monte Carlo simulation methods to generate spin configuration of the XY model involving a mix of the Wolff cluster update and the local Metropolis update [36, 39]. For the training set, we generate 1000 configurations per temperature, with 64 temperatures ranging from 0.1 to 3.0, for lattice sizes $L = 8, \ldots, 64$ in increments of 8. The test set is generated separately, with 100 configurations per temperature. From our generated samples, we find $T_{\rm KT} = 0.899 \pm 0.06$, which is consistent with the literature value of $T_{\rm KT} = 0.893$ [79, 81]. As shown in Fig. 2.3b, the magnetization evaluated at the critical point, $M|_{T_{\rm KT}}$, is of significant magnitude, and scales with $L^{-1/8}$ as expected [79], to within a 4% error.

2.3 Classification of phases

We study the binary classification of the two phases of the XY model, labeling configurations as belonging to either the low $T < T_{\mathrm{KT}}$ or high $T > T_{\mathrm{KT}}$ temperature phases. Our goal is to confirm whether simple supervised learning with neural networks is capable of correctly classifying spin configurations according to these labels. In particular, we wish to interpret whether the network relies on the (finite-size) magnetization, or on topological defects. Further, we inquire as to what specific network architecture is required to achieve this goal and what features different architectures may utilize.

From the training data, we randomly select 10% for cross-validation, in order to decrease the chance of overfitting and to identify a definitive stopping point for training using early stopping [84].

The network is trained to minimize the loss function $L(y^{\mathrm{pred}}, y^{\mathrm{true}})$, where y^{true} represents the true binary labels and y^{pred} the predicted ones. We take the loss function to be the standard cross entropy

$$L(y^{\mathrm{pred}}, y^{\mathrm{true}}) = -\sum_i y_i^{\mathrm{true}} \log y_i^{\mathrm{pred}} \,. \tag{2.4}$$

The parameters of the network (weights and biases) are then optimized through back-propagation to minimize the loss function on the training data [16]. Each network is trained until the loss function *evaluated on the validation set* fails to decrease after 50 training epochs. Early stopping with cross-validation is commonly used to choose the network parameters with minimal generalization error [84]. We implement the networks with the Keras library using the TensorFlow backend [85, 86].

We employ two different standard network architectures: a one-layer feed-forward network (FFNN) and a deep convolutional neural network (CNN). The FFNN consists of one hidden layer of 1024 sigmoid activation units and one sigmoid output unit. The CNN starts with a two-dimensional convolutional layer consisting of 8 filters of size 3×3 with rectified linear unit (ReLu) activation functions. The output from this layer is passed to another identical convolution layer with 16 filters before applying 2×2 maxpooling. The network is then reshaped and fed into a fully connected layer with 32 ReLu units and passed to a single sigmoid output unit. Because there is a total of $1024\,L^2 + 2049$ trainable parameters in the FFNN, it can be difficult to train as compared to the $128\,L^2 - 1024\,L + 3361$ parameters in the CNN. The network architectures are listed in Table 2.1. This is because the CNN explicitly takes advantage of the two-dimensional structure of the input to vastly improve performance. This architecture is one of the simplest that can attain over 99% accuracy on the standard MNIST dataset. In our experience, changing the hyperparameters had negligible effect on the accuracy.

Network	FFNN	CNN

Layers	Dense(1024)
	σ
	Dense(1)
	σ
		Conv(8, (3,3))
		ReLu
		Conv(16, (2,2))
		ReLu
		MaxPooling
		Dense(32)
		ReLu
		σ(Dense(1))

Parameters	$1024\,L^2 + 2049$	$128\,L^2 - 1024\,L + 336$

Table 2.1: Network architecture for binary classification of phases.

One goal in modern machine learning is to minimize the amount of feature engineering required. In our case, this corresponds to treating the raw spin configurations as direct inputs to the neural networks. For the XY model, this data is formatted as angle values, $\theta \in [0, 2\pi)$, on an $L \times L$ lattice with periodic boundary conditions.

For a given spin configuration, the output value of the final neuron in the network gives the probability of the configuration belonging to the low- (or high-) temperature phase. In Fig 2.4, we show the output probability of each phase for a CNN trained on $L = 16, 24, 32$ data. This is directly analogous to Fig. 1.7 from Chapter 1. As system size increases, the becomes slightly sharper, although not as quickly as in the Ising case.

Due to thermal fluctuations, it is difficult to accurately classify states near the critical point. In accordance with intuition about phase transitions, we take the point where the probability is 0.5 to be the inferred critical temperature $\widetilde{T}_{\text{KT}}$. This is further established in Ref. [60] where the authors show that this point scales with the correct correlation length critical exponent and predicts the thermodynamic critical temperature accurately for the Ising model. In that case, training a FFNN with a single hidden layer of 100 sigmoid units was sufficient ($100\,L^2 + 202$ total parameters) to achieve high classification accuracy and correctly predict the critical temperature.

Similarly, we study the performance of both a FFNN and a CNN in predicting T_{KT} for the XY model. To get an estimate for the statistical variance, the training process is repeated ten times with different validation sets.

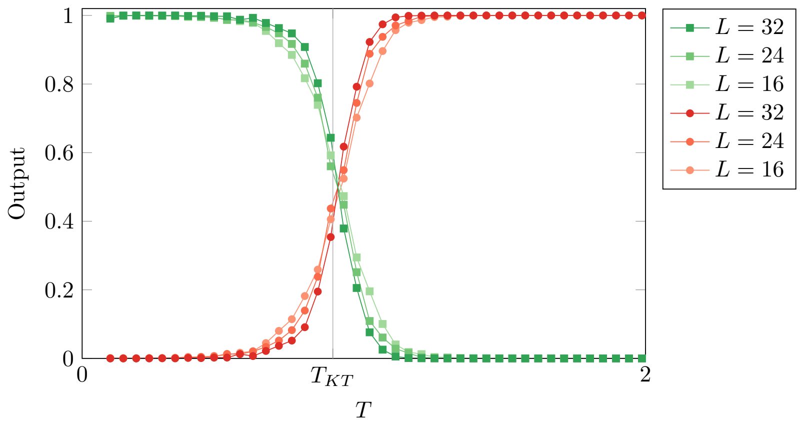


Figure 2.4: Probability of each phase for a CNN trained on raw spin configurations. The green lines are the probability to be in the quasi-long range order phase. Analogous to Fig. 1.7 from Chapter 1, Section 1.2.4.

As illustrated in the inset of Figure 2.5a, the FFNN has low classification accuracy (i.e., percentage of correctly classified configurations) for $L > 48$. This results in the very poorly predicted critical temperature, $\tilde{T}_{\mathrm{KT}}$, in the main plot. In contrast, the accuracy of the CNN continually improves as L increases. However, as evident from Fig. 2.5, there is no clear finite-size scaling trend in the predicted T_{KT}. To interpret this, we note that for each system size, the network is supervised on the thermodynamic value of T_{KT}. Thus, we speculate that each network could simply be learning to discriminate phases based on a robust, global feature that distinguish the regions above and below T_{KT} for any L.

Based on previous experience, a global magnetization is a feature very easily detected in a supervised learning scheme [60, 70, 87]. Since the finite-size configurations of the XY model themselves contain a non-zero magnetization at $T > 0$ (see Fig. 2.3b), it is reasonable to hypothesize that the CNN simply learns this threshold value of the magnetization for each system size separately. Because of the Mermin-Wagner theorem, however, it is known that a global magnetization is not a relevant feature for T_{KT} in the thermodynamic limit. Thus, in this case, some amount of feature engineering is crucial to achieve our goal of detecting a phase transition mediated by topological defects.

In the next step, we preprocess the spin configurations into the associated vorticity and train the networks on these configurations. To calculate the vorticity, one computes the angle differences $\Delta\theta_{ij} \in [-2\pi, 2\pi]$ between each pair of neighbouring spins i and j on a plaquette and converts these to the range $(-\pi, \pi]$. This can be done by applying the

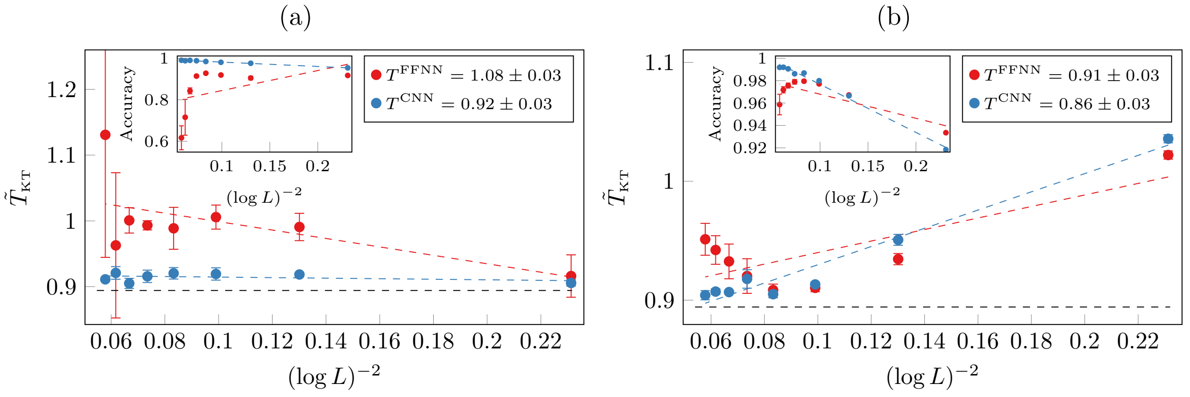


Figure 2.5: Finite-size scaling of the predicted T_{KT} for FFNN and CNN trained on either (a) raw spin configurations, or (b) the vorticity. In either case the FFNN performs worse than the CNN according to the test classification accuracy (insets). The critical temperature is determined by the point where the sigmoid output, as a function of temperature, crosses 0.5. Each data point and variance is obtained by training 10 networks with stochastic gradient descent until the validation loss function fails to improve after 50 epochs (early stopping).

sawtooth function,

$$\mathrm{saw}(x) = \begin{cases} x + 2\pi, & x \leq -\pi, \\ x, & -\pi \leq x \leq \pi, \\ x - 2\pi, & \pi \leq x, \end{cases} \tag{2.5}$$

to each $\Delta\theta_{ij}$. The sum of the rescaled angle differences gives the vorticity from Eq. (2.2).

Trained on the vortex configurations, Fig. 2.5b shows that both the FFNN and CNN achieve high accuracy and scale with L towards the correct value of T_{KT}. However, once again we observe that the FFNN begins to perform poorly for $L > 32$, whereas the CNN continually improves. We note that the scaling seems consistent with Eq. (2.3), particularly for the CNN. However, from this scaling alone, we cannot determine precisely what the CNN learns. For example, it could potentially classify the phases based on the sum of the squared vorticity (which is approximately zero below T_{KT}), or it might represent a more complicated function such as the average distance between vortex-antivortex pairs. Regardless, the scaling behaviour may serve as a useful diagnostic to determine whether a given network is learning bulk features or topological effects.

2.4 Learning by confusion

We further investigate the difference between training on spin configurations and vortex configurations by employing a confusion scheme [77, 88]. Learning by confusion offers a semi-supervised approach to finding the critical temperature separating two phases by training many supervised networks on data that is deliberately mislabelled. The binary label '0' is assigned to a configuration if its temperature is less than a proposed T^* and '1' otherwise. A new network is trained on each new labeling of the data, (i.e., for each T^*). It is expected that the highest accuracy is achieved when the labeling is close to the true value, and, trivially, at the end points. This results in a $\vee\!\vee$ shape when plotting the test accuracy as a function of T^* [77]. The peak on either endpoint can be attributed to the network being trained and tested exclusively on one class, in which case it will always place test data into that class. The key assumption in the confusion scheme is the existence of a true physical labeling of the data which the network is capable of learning more accurately than false labellings.

Since we have shown that the CNN is more successful at classification than the FFNN, we only consider the CNN for the present confusion scheme. The results of training on raw spin and vortex configurations are shown in Fig. 2.6. Learning on the raw spins results in a $^{-}\!\vee$ shape rather than the expected $\vee\!\vee$. As mentioned above, the finite-size XY model has a non-zero magnetization for $T < T_{\mathrm{KT}}$ and this algorithm can easily classify any division $T^* < T_{\mathrm{KT}}$ by a threshold magnetization. This supports our hypothesis from Section 2.3 that trained on raw spins, a CNN learns the magnetization.

When trained on vortices, the expected $\vee\!\vee$ shape emerges, although it is skewed because we choose our training data from a non-symmetric region around T_{KT}. Despite having a powerful deep network, it is unable to learn any arbitrary partition and performs best near T_{KT}. This may be attributed to the fact that for low T, the vortex configurations are fundamentally similar; there are few vortices and they are logarithmically bound. This is in contrast with the raw spin configurations which may possess distinguishing features like the magnetization. Near T_{KT}, the network can distinguish the phases with high accuracy because of the true physical partition due to vortex unbinding. At high T^* the vortex configurations look sufficiently random that the network again misclassifies for an arbitrary partition.

We also observe significant finite-size effects in the $\vee\!\vee$ and $^{-}\!\vee$ shape, both broadening and shallowing with increasing L. The finite-size scaling behaviour of the peak does not trend towards T_{KT} in the vortex case, but rather always stays above it, similar to the specific heat peak (see Fig. 2.6c). Surprisingly, in Fig. 2.6c, we see the confusion scheme

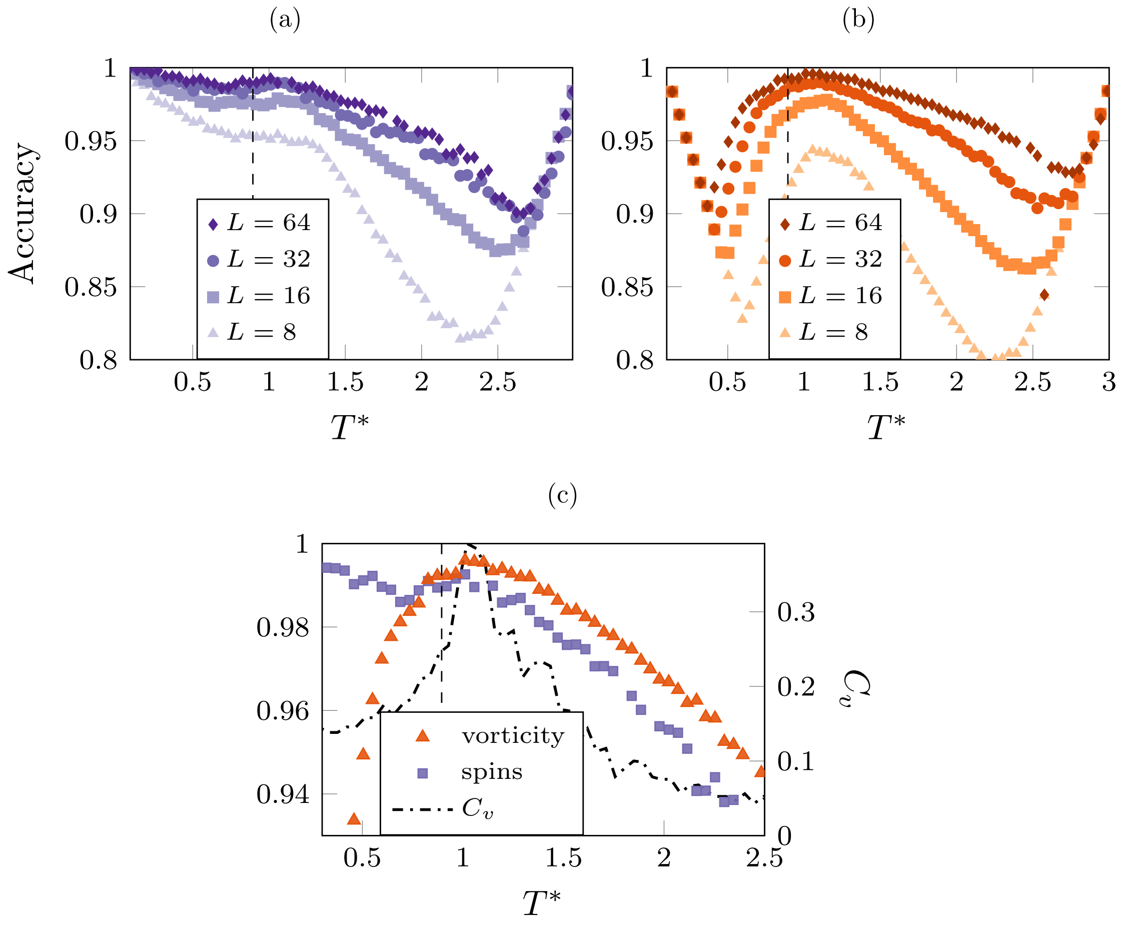


Figure 2.6: The learning by confusion scheme for a CNN applied to: (a) raw spin configurations, (b) vorticity configurations. The test accuracy is expected to form a W shape with the peak at $T^* = T_{\mathrm{KT}}$. In (c), the peak in specific heat (C_v) is compared to the peak of the test accuracy for a system of size $L = 64$. The dashed vertical line s1hows the thermodynamic T_{KT}.

achieves higher accuracy at $T^* \approx 1$ than $T_{\text{KT}} = 0.89$, which indicates that the false $T^* \approx 1$ phase boundary is easier for the network to learn than the temperature $\tilde{T}_{\text{KT}}$ predicted by the universal jump. While this effect might disappear in the thermodynamic limit, it is still troubling. Matters are even worse for training on raw spins since all $T^* < T_{\text{KT}}$ have accuracy greater than 98.5% for $L = 64$, so it is even unclear where $\tilde{T}_{\text{KT}}$ is.

For finite-size systems, the test accuracy curve will never go flat. The reason is that a single spin configuration does not unique belong to a particular temperature, but rather it occurs probabilistically for all temperatures (although perhaps infinitesimally). This always will result in some classification error. In particular, for isotropic or highly thermal regions, it is impossible to accurately correctly classify states, and therefore a $\vee$ shape occurs. For other regions, the curve will go flat in the thermodynamic limit. Interestingly, the confusion scheme inadvertently tells us information about the variances in possible temperatures of a state.

The confusion scheme for the XY model offers insight into what our CNN prefers to learn. In the case of the raw spin configurations, we infer that it learns the finite magnetization of the spin configurations instead of topological features. Near T_{KT}, the network trained on vortices achieves slightly higher accuracy (see Fig. 2.6c); therefore, in this case, the network would benefit from learning vortices. Despite this argument, we stress that we have no strong evidence that our CNN is even capable of finding vortices. To address this, in the next section we propose a custom network designed for vortex detection and test if it works in practice.

2.5 Supervised learning of vortices

Training a neural network to recognize vortices is different from binary classification. Instead of a single number (or word) labeling the image, we have an entire array of numbers w, where each number corresponds to one plaquette of the lattice.

We train a supervised CNN implemented in TensorFlow to recognize vortices. The input to the network is spin configurations on a square lattice generated by Monte Carlo sampling and the labels are created by explicitly calculating the winding numbers for each square of spins using Eq. (2.2). The label for classification is no longer a single number, but rather a two-dimensional array of numbers, with values of $+1$ (vortex), -1 (antivortex), and 0 otherwise.

Instead of training directly on the vorticity, it helps to split the label into three channels. This is the equivalent of one-hot encoding vectors into binary vectors, except applied to

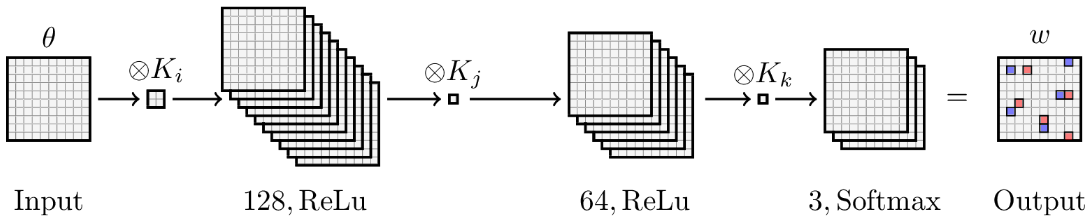


Figure 2.7: Network architecture for supervised learning of vortices. On the input spins we apply 128 convolution filters of size 2×2 to capture interactions between spins. After applying ReLu activation functions, the next layer is 64 filters of size 1×1, again with ReLu activations. The network outputs three binary channels with a softmax activation function to ensure only one label is associated to each square in the lattice. Each channel represents ones of the possible values of the winding number.

matrices. Instead of the vorticity $w \in [0, \pm 1]$, we rewrite this as three binary arrays containing only 0's and 1's. To revert to the ordinary 'one-channel' vorticity, we simply subtract the $w = -1$ channel from the $w = +1$ channel.

Figure 2.7 displays the full network architecture. The motivation for this structure is that the first convolutional layer with 2×2 filters might learn local angle differences. The three-channel output is activated with a softmax function which forces the network to choose only one value for the vorticity of each plaquette.

In Fig. 2.8, the loss function during training is shown for different lattice sizes. Each network was trained using the Adam optimizer and early stopping with a patience of ten epochs to prevent overfitting. It turns out that this network architecture readily achieves over 99% accuracy in identifying vortices.

Adding L2 regularization had no effect on the performance of the network; however, ReLu units were substantially better than tanh or sigmoid functions. Likewise, a single-layer fully-connected network did not perform well, yet convolutions layers worked perfectly. Increasing the number of layers or neurons resulted in faster convergence.

Extending this to the quantum realm has already been done for 1D topological band insulators by Zhang et al. [76]. In this case, the label is not an array, but rather a single number (the global winding number), which describes the topological sector of a Hamiltonian. With a network similar to our Fig. 2.7, Zhang et al. achieve very high accuracy and can even detect higher-order winding numbers not included in the training data.

We have demonstrated that a neural network can easily learn to recognize vortices when trained on label data, yet it remains unclear if an unsupervised method could achieve

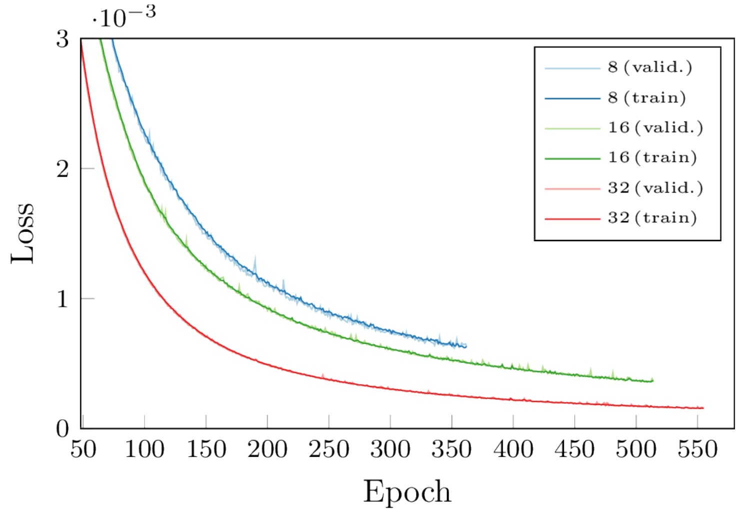


Figure 2.8: Training and cross-validation loss function for system sizes from 8×8 up to 32×32. Training is stopped once the loss on the cross-validation set fails to decrease after ten epochs.

similar results. So far, the results from PCA and variational autoencoders both suffer from learning the energy or magnetization instead of vorticity [71, 89, 74]. This is perhaps not surprising since, while in the thermodynamic limit the 2D XY model has no magnetization, a finite-size system has a very large magnetization in the low-temperature region. Even a lattice the size of Texas would have a significant magnetization [90]. While theorists can take the continuum limit to avoid these finite-size effects, machine learning algorithms don't have this privilege and must work with only the data they are given.

2.6 When is it beneficial to learn vortices?

In the previous sections, we compared networks trained on the raw spin configurations to those trained on vortex configurations that were constructed manually (i.e., feature-engineered). We further demonstrated that is it possible for a custom CNN network to learn vortices. We now explore the possibility of a network architecture designed specifically for learning vortices as an intermediate representation, before performing classification.

It is one of the remarkable features of deep neural networks that each layer may represent a new level of abstraction [91, 92, 93]. For example, in facial image recognition, the first convolution layer may extract edges, while the final layer encodes complex features such as facial expressions [94]. We aim to design a network which may similarly be interpreted as representing vortices in an intermediate layer.

Below, we derive the appropriate weights for a three-layer network which computes the vorticity from input spin configurations. The entire network is visualized in Fig. 2.9.

The first layer, which acts on the input angle values, θ_i, is a convolution layer with four 2×2 convolution filters given by

$$K_1 = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}, \qquad K_2 = \begin{bmatrix} 0 & -1 \\ 0 & 1 \end{bmatrix},$$
$$K_3 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, \qquad K_4 = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}.$$

The effect of these filters is to compute the nearest neighbour angle differences, $\Delta\theta_{ij}$, within each plaquette. In the next layer, we apply is hard-coded to map the angle differences, $\Delta\theta_{ij} \in [-2\pi, 2\pi]$, into the range $[-\pi, \pi)$. This is done by applying the sawtooth function from Eq. (2.5) to each element in the $(L, L, 4)$-dimensional array. The final processing layer computes a weighted sum of the four angle differences by applying a single 1×1 convolution filter. Uniform weights with zero biases would compute the vorticity exactly up to a multiplicative constant.

While the network described above is capable of representing vortices within an internal layer (vorticity layer in Fig. 2.9), it might fail to do so in practice. To explore this we consider three possible variations of the initializations of the network parameters.

The first variation consists of fixing the weights (and biases) in the first three layers such that the network computes the vorticity exactly. This is, of course, engineering the relevant features by hand; however, it provides a useful benchmark. The second variation is performed by initializing the weights exactly to those of the fixed network, then relaxing the constraints as training is continued. This step shows whether the original (vortex) minimum is stable. The third variation is simply the naive choice where the network parameters are initialized randomly.

For all three variations, we train for binary classification by minimizing the cross-entropy loss from Eq. (2.4). Each network is trained 10 times with different validation sets. As per Section 2.3, we implement early stopping to terminate training once the loss function on the validation set fails to improve after 50 epochs. We train on lattice sizes from $L = 8, \dots, 72$ in increments of eight.

We can understand the three variations by looking at the loss function evaluated on the test set as in Fig. 2.10. For small L, the loss function of the fixed network is much larger

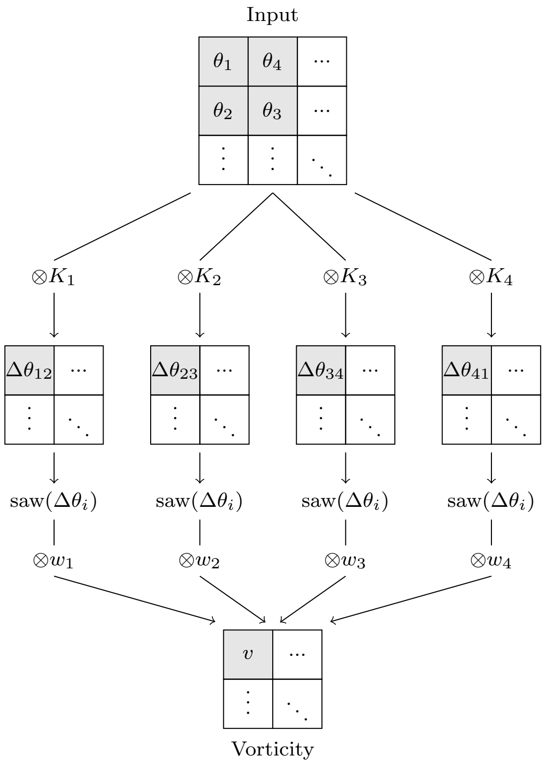


Figure 2.9: Visual representation of how the custom network architecture can compute the vorticity. We denote the convolution operation with $\otimes$, and ignore biases for the purpose of the diagram. Applying the four 2×2 filters, K_i, partitions the data into four $L \times L$ arrays where each element is an angle difference in one lattice direction, $\Delta\theta_{ij}$. The angle differences are then converted into the range $\Delta\theta_{ij} \in [-\pi, \pi)$ by applying the sawtooth function from Eq. (2.5). A single 1×1 convolution filter with weights $w = [1, 1, 1, 1]$ and zero biases then sums the four shifted angle differences into the vorticity.

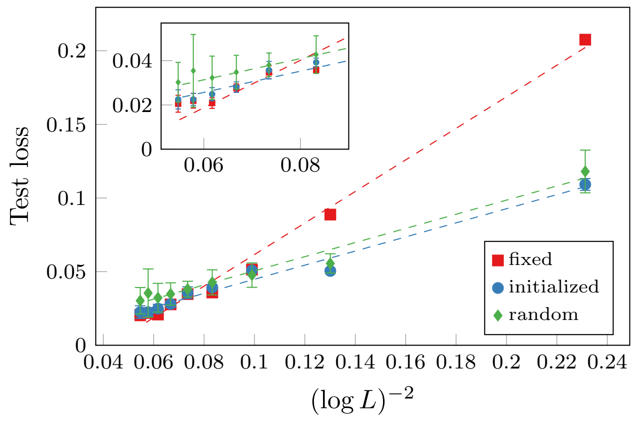
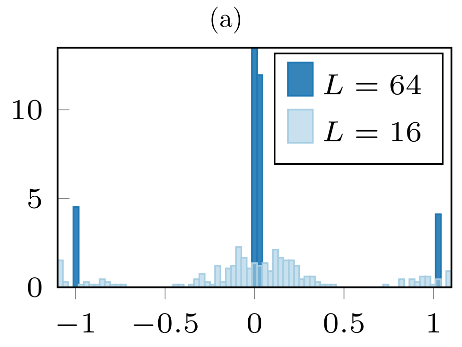


Figure 2.10: The loss function from Eq. (2.4) evaluated on the test set for three variations of the custom architecture for various lattice sizes L. For small $L < 16$, the fixed network with hard-coded weights performs poorly compared to the others. For large $L > 32$, the fixed network performs best, possibly due to a reduced number of trainable parameters. The inset shows a magnified region for $32 \leq L \leq 72$.

than the others, indicating that it is *not* beneficial to represent the vortices for $L < 16$. In this small-lattice region, learning vortices hinders classification. However, near $L \sim 32$ the fixed network outperforms the other two. Hence, we conclude that only for the large-lattice region, $L > 32$, is it beneficial for a network to learn an intermediate representation of the vorticity. This also agrees with the findings in Ref. [71] in which the topologically-invariant winding number could be learned for systems of size $L > 32$.

We can check what each network learns by looking at the histogram distribution of the outputs of the vorticity layer in Fig. 2.9. For the fixed network, we would see exactly integral quantities corresponding to the quantized vorticity. For the vortex-initialized network, Fig. 2.11 shows that for small L, it does not learn the true vorticity distribution, but for $L \geq 32$ it does. This is consistent with the hypothesis that learning vortices is only beneficial for $L > 32$. The randomly initialized network does not produce a histogram consistent with the learning of vortices for any system size studied.

Interpreting the behaviour of the neural network for large L is not straightforward. As Fig. 2.10 shows, the model with fixed features and fewer trainable parameters performs better for large L. This can likely be attributed to a lower-dimensional optimization landscape. We cannot conclude whether the vortex representation is a global minimum for the fully adjustable (randomly initialized) network variation. While it certainly performs best in fixed computational time, the higher dimensionality of the adjustable network may



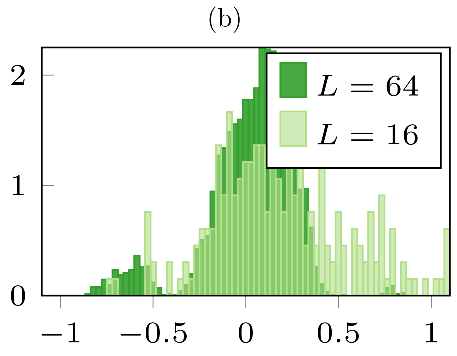


Figure 2.11: Histogram of the values of the vortex layer from Fig. 2.9 which (ideally) computes the vorticity for: (a) the network initialized to compute vorticity, and (b) the randomly initialized network. In (a), we see for small L, the vorticity is not quantized, indicating that the network did not learn to compute the local vorticity; however, for large L, the histogram looks as expected for vortex detection. Conversely, the distribution in (b) appears unrelated to vortices for any L.

have another global minimum not present in the lower-dimensional case. We can claim, however, that the vortex minimum is at least a stable local minimum since a network initialized to it never escapes, as demonstrated by the initialized variation for large L in Fig. 2.10.

Adding a custom regularization term could potentially alter the optimization landscape to aid the network in detecting vortices. One method would be to enforce integral quantities for an intermediate output of the network, but in our attempts, this results in the intermediate quantity peaked sharply around zero. There is also the possibility of adding a regularization to the initial kernels to learn only nearest neighbours interactions, but this is overly restrictive and defeats the purpose of automated machine learning.

2.7 Conclusion

In this Chapter, we asked whether it is possible for a neural network to learn the vortex unbinding at the KT transition in the two-dimensional classical XY model. We demonstrated the significant effects that feature engineering and finite lattice sizes have on the performance of supervised learning algorithms.

Treating spin configurations as raw images and training on the thermodynamic value for T_{KT}, we found that naive supervised learning with a feed-forward network failed to converge to an accurate estimate for the KT transition temperature for moderate lattice sizes ($L \approx 32$). Conversely, a convolutional network performed consistently well with increasing L. Since the prediction of T_{KT} from the convolutional network was insensitive to L, we inferred that the network extracted features related to the magnetization, which are present in any finite-size lattice. This conclusion was further supported by the observation that in the confusion method any false phase boundary T^* below T_{KT} could easily be learned by a network when trained on the raw spin configurations.

By preprocessing the spin configurations into vorticity, both network architectures displayed finite-size scaling behaviour consistent with the thermodynamic value of T_{KT}. In particular, the performance of the convolutional network continually improved as the system size increased, whereas the one-layer network's performance plateaued around $L = 32$. When the confusion scheme was trained on vortices it did not predict the correct critical temperature; instead, the test accuracy reached a maximum near $T^* \approx 1$. This demonstrates the need for further study of the confusion scheme for the semi-supervised learning of phase transitions.

We further explored if such extreme feature engineering could be relaxed while retaining acceptable accuracy. We devised a deep-layered structure of weights that could be constrained to extract vortices from the raw spin configurations or left free to explore other minima in the learning process. Although is it possible to learn vortices, the network can also perform its classification task to reasonable accuracy by finding a local optimization minimum that is unrelated to topological features. We found that it is beneficial for the network to discover vortices only for lattices with of over 1000 spins. Yet, even for large system sizes, a randomly initialized network settled into a local minimum not related to vortices.

It is likely that the optimization landscape of our designed network is sufficiently rough so that stochastic gradient descent would take exponentially long to find a minimum where the learned features correspond to vortices.

The difficultly that these standard supervised learning techniques have in discriminating the phases of the XY model underscores the challenge that unsupervised learning techniques could face in learning the KT transition from unlabelled data. Our work emphasizes the need for further study into how much feature engineering is required before topological features can be used reliably for the machine learning of unconventional phases and phase transitions.

3

Super-Resolving and Renormalization Group

The renormalization group (RG) is of profound important in physics. It describes an example of low-energy physics emerging from high-energy degrees of freedom. This mirrors what happens in deep learning, where the numerous parameters in a deep neural network interact such that they can solve a high-level problem such as image classification. It was recognized early that deep learning may even be a more general form of the RG [95]. This Chapter provides a concrete example of learning to invert the RG procedure in a bold attempt to infer high-energy physics from statistical patterns. The majority of this Chapter appeared in [3].

3.1 Introduction

A primary challenge in the field of quantum many-body physics is the efficient computational simulation of systems with a large number of particles. Such simulations are crucial for the investigation of strongly-interacting systems, the discovery of exotic phases of matter, and the design of new quantum materials and devices. Recently, it has been proposed to treat the many-body problem as data-driven, whereby the large dimensionality of the data motivates the adoption of machine learning algorithms [60, 13].

In condensed matter systems, neural networks were first used as supervised classifiers that distinguish phases and identify phase transitions, even in unconventional cases when there is no underlying order parameter [60, 96, 70, 1]. Furthermore, unsupervised generative models have been shown to successfully capture thermal distributions [97, 98, 70]. In the quantum case, neural networks are being employed as representations for many-body wave-functions, with broad applications such as variational ansatz [99], guiding functions [100], or for quantum state tomography [101, 102, 5].

Early connections between statistical physics and machine learning drew parallels to the renormalization group [103, 104] (RG), a canonical paradigm in physics that involves iteration through a series of coarse-graining and rescaling procedures. The mathematical similarity of the RG procedure to the processing of information in multi-layer neural networks has driven interest in examining the theoretical underpinnings of deep learning [105, 95]. Conversely, numerous relations between the RG and machine learning have proven useful for physics itself where neural networks have been proposed as generative models that assist RG procedures [106], or for identifying relevant degrees of freedom to decimate [107, 108]. Furthermore, direct applications of neural networks on physical configurations can produce RG or inverse RG flows [108, 109, 110, 106].

Outside of physics, an area of expanding application for machine learning is image super-resolution, where the goal is to increase the number of pixels in an image while (subjectively) maintaining the perceivable quality [111, 112]. Remarkable progress has been made with convolutional neural networks (CNNs), which can be used to reconstruct high-resolution images to photo-realistic quality [112].

In this Chapter, we investigate whether super-resolution methods may be useful in condensed matter and statistical physics by allowing one to produce lattice configurations of larger sizes directly from those obtained for smaller systems. For concreteness, we focus on the classical Ising model in one and two dimensions. Our method takes a configuration of Ising spins on a lattice and subjects it to a majority rule block spin RG procedure [113]. A CNN is then trained to invert this transformation by being exposed to both the higher and lower resolution lattices. Since some information is necessarily lost in the RG step, the network output is interpreted as a probabilistic image that is sampled to produce super-resolved images.

We give numerical evidence that the trained super-resolution network performs a probabilistic inverse of the RG transformation and reproduces thermodynamic quantities on larger lattices starting only from smaller ones. In addition, we propose a way to extrapolate the weights of a trained CNN to apply it to sizes larger than those available in the training data. Using this idea iteratively, we acquire configurations of increasing size that we use to estimate the critical exponents of the 2D Ising universality class, obtaining agreement with known theoretical results.

3.2 Super-resolution and RG

Super-resolution is defined as a mapping $\mathcal{SR} : \mathbb{Z}_2^{L\times L} \to \mathbb{Z}_2^{fL\times fL}$ from a low-dimensional space of $L \times L$ images to a high-dimensional space of $fL \times fL$ images where $f > 1$ is the upscaling factor. We use $\mathbb{Z}_2$ to denote a binary variable, with $\mathbb{Z}_2^{L\times L}$ denoting an $L \times L$ matrix of binary values. The objective of image super-resolution in computer vision is to achieve a high perceived quality on the super-resolved image. This is generally a subjective criterion. To give a more quantitative definition, quantities like peak signal-to-noise ratio (PSNR) or structural similarity (SIM) have been used [111, 112]. However, even these quantities might not accurately estimate the quality perceived by a human [112].

In contrast, statistical physics provides a well-defined objective for super-resolution of physical systems since super-resolved configurations should follow a specific statistical ensemble. Basic thermodynamic quantities like the magnetization or energy serve as an indicator of whether a super-resolved image is consistent with this ensemble.

We proceed by reviewing the real-space decimation of the Ising model before discussing the relevant network architecture.

3.2.1 Decimation of the Ising model

Let $\boldsymbol{\sigma} \in \mathbb{Z}_2^{2L\times 2L}$ be an Ising configuration of N spins that follows the Boltzmann distribution

$$P_K(\boldsymbol{\sigma}) = \frac{1}{Z} e^{-H(\boldsymbol{\sigma})/T} = \frac{1}{Z} e^{K \sum_{\langle ij \rangle} \sigma_i \sigma_j}, \tag{3.1}$$

where $K = 1/T$ and $Z(K) = \sum_{\{\boldsymbol{\sigma}\}} e^{-H(\boldsymbol{\sigma})/T}$ is the partition function. We take periodic boundary conditions (PBC) and include only nearest-neighbour interactions throughout this Chapter.

Consider the real-space coarse-graining of the 2D Ising model according to the majority rule [103]. A $2L \times 2L$ lattice is divided into 2×2 blocks where each block is transformed to a spin with the same state as the majority of spins in the block. If the total sign is zero, we take the sign of the upper left spin to make the procedure deterministic, instead of the more common probabilistic approach of taking a random sign [38].

A low-resolution configuration, $\mathbf{s} = \mathcal{MR}(\boldsymbol{\sigma}) \in \mathbb{Z}_2^{L\times L}$, is obtained upon applying the deterministic majority rule. Such a configuration follows the marginalized distribution:

$$\tilde{P}_{\tilde{K}}(\mathbf{s}) = \sum_{\{\boldsymbol{\sigma}\}} k(\mathbf{s}, \boldsymbol{\sigma}) P_K(\boldsymbol{\sigma}), \tag{3.2}$$

where $k(\mathbf{s}, \boldsymbol{\sigma})$ is the kernel of the transformation [114]. In this way, the distribution of low-resolution configurations, $\tilde{P}_{\tilde{K}}(\mathbf{s})$, is directly related to the distribution of high-resolutions $P_K(\boldsymbol{\sigma})$. Evidently, any super-resolution procedure must satisfy the identity relation

$$\mathcal{MR}(\mathcal{SR}(\mathbf{s})) = \mathbf{s}\,. \tag{3.3}$$

A stronger requirement, and the main challenge, is to discover a map such that $\mathcal{SR}(\mathcal{MR}(\boldsymbol{\sigma}))$ obeys the correct Boltzmann distribution. We emphasize that $\mathcal{SR}(\mathcal{MR}(\boldsymbol{\sigma}))$ is not necessarily equal to $\boldsymbol{\sigma}$, since only the distributions need to match, not each individual configuration. Furthermore, in our majority rule decimation, we have not rescaled the Hamiltonian couplings as required in the conventional definition of a complete RG step. It may be possible to learn the rescaling with a neural network; however, in this Chapter we simply fit the couplings as needed for a consistent rescaling by using a numerical approximation.

The overall procedure can be visualized in Fig. 3.1. Starting with a configuration of system size L, we decimate to a smaller size $L/2$ denoted by $\mathcal{MR}(\boldsymbol{\sigma}_L)$.

$$\begin{tikzcd}
 & \mathcal{S}R(\boldsymbol{\sigma}) \\
\boldsymbol{\sigma}_L \arrow[ur, "\mathcal{S}R"'] \arrow[r, leftrightarrow, "\approx"'] \arrow[d, "\mathcal{MR}"'] & \mathcal{S}R(\mathcal{MR}(\boldsymbol{\sigma}_L)) \\
\mathcal{MR}(\boldsymbol{\sigma}_L) \arrow[ur, "\mathcal{S}R"'] &
\end{tikzcd}$$

Figure 3.1: Schematic of the super-resolution procedure. Starting with a configuration $\boldsymbol{\sigma}$ generated via Monte Carlo (MR) at size L, we apply the majority rule ($\mathcal{MR}$) decimation to a downsampled (DS) configuration. The map $\mathcal{SR}$ consists of two steps; upsampling (F_θ) and rescaling (f^{-1}). We assume that the map $\mathcal{SR} = f^{-1} \circ F_\theta$ is learnable and applies to any input size.

3.2.2 Network architecture

We now attempt to invert the majority rule procedure using a supervised learning approach. The unknown super-resolution mapping $\mathcal{SR}$ is parametrized with a deep convolutional neural network (CNN). CNNs are ideal for our problem due to their utility in image processing tasks. Moreover, the weight-sharing property of convolutions allows extrapolation to larger sizes.

The first layer of our network is an upsampling layer that increases the resolution from $L \times L$ to $2L \times 2L$ by transforming each up (down) spin to a block of four up (down) spins (Fig. 3.2a). At very low temperatures, configurations are fully polarized, so we expect this upsampling to be highly accurate. However, non-fully-polarized blocks appear at higher temperatures, making naive upsampling insufficient. In order to alleviate this, the convolution layers that follow must add the required statistical fluctuations, similar to the Monte Carlo sweep procedure in Ref. [113].

Each convolution layer takes a configuration $\mathbf{x} \in \mathbb{R}^{2L\times 2L}$ as input and applies the transformation $f(W * \mathbf{x} + \mathbf{b})$, where $*$ denotes the convolution operation, W is the so-called filter, $\mathbf{b}$ is a bias vector and f is a non-linear differentiable function applied element-wise (Fig. 3.2b). This function is known as an activation function and the typical choice is the rectified linear function $\mathrm{ReLU}(x) = \max(0, x)$. The effect of each convolution is to combine local features within a $n_f \times n_f$ region (filter size). A consequence of this is that each convolution layer reduces the image size by eliminating the right-most and bottom edges. To avoid truncating the image edge, we surround the original configurations with additional spins from the periodic boundary conditions (Fig. 3.2b). This has the advantage of respecting the boundary conditions of the underlying physical model.

In order to obtain a discrete Ising configuration, the sigmoid activation $\sigma(x) = 1/(1 + e^{-x})$ is used in the final layer, giving an output $\mathbf{p} \in [0, 1]^{2L\times 2L}$. This is treated as the probability that the corresponding spin is up. A discrete configuration is then obtained by sampling $\mathbf{p}$ for each lattice site (Fig. 3.2c).

The network is characterized by parameters θ, which include all the weights, W, and biases, $\mathbf{b}$, from each convolution layer. These are tuned to minimize a loss function defined on a dataset of inputs $\mathbf{s}_i \in \mathbb{Z}_2^{L\times L}$ and targets $\boldsymbol{\sigma}_i \in \mathbb{Z}_2^{2L\times 2L}$ with $i \in \{1, 2, \ldots, n\}$, where n is the number of samples in the dataset. In contrast to typical supervised learning applications (e.g. handwritten digit recognition), the dimensionality of the output is larger than the input. The loss quantifies the distance between predicted output $\mathcal{SR}_\theta(\mathbf{s}_i)$ and the original high-resolution $\boldsymbol{\sigma}_i$. Minimization is done with back-propagation which involves calculations of gradients, and thus cannot be done using the final sampled (discrete) output.

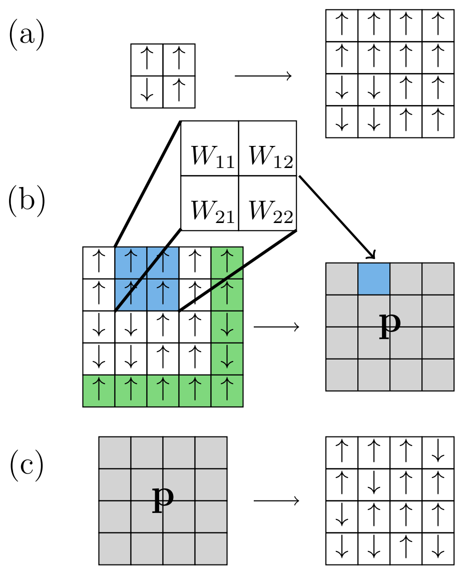


Figure 3.2: (a) Upsampling by replacing each up (down) spin with a block of four up (down) spins. (b) The weights W convolve local regions together and add a bias b_i. Applying a sigmoid function element-wise gives the probabilities of each site being up as $\mathbf{p}$. Green sites correspond to the PBC padding. (c) Sampling $\mathbf{p}$ gives discrete Ising spins on the super-resolved $2L \times 2L$ lattice.

To be consistent with the interpretation of the continuous outputs $\mathbf{p}$ as probability we use the cross-entropy loss function;

$$L(\{\boldsymbol{\sigma}_i\}, \{\mathbf{p}_i\}) = -\sum_{i=1}^{n} \left[\boldsymbol{\sigma}_i \cdot \ln \mathbf{p}_i + (1 - \boldsymbol{\sigma}_i) \cdot \ln (1 - \mathbf{p}_i)\right], \tag{3.4}$$

where $i \in \{1, 2, \ldots, n\}$ and $\cdot$ denotes the element-wise product between matrices. Note, we are free to add additional terms to Eq. (3.4) to assist training. For example, as we will see, it is sometimes beneficial to introduce a term proportional to $|E(\boldsymbol{\sigma}_i) - E(\mathbf{p}_i)|^2$.

3.2.3 Extrapolation to larger lattices

We approximate the super-resolution mapping $\mathcal{SR}$ as a neural network $\mathcal{SR}_\theta$ with parameters θ. In order to train the network $\mathcal{SR}_\theta$, we need access to $2L \times 2L$ configurations. We obtain these with Monte Carlo simulations. In this sense, the method does not allow us to access sizes larger than the ones we have already simulated. It would be useful if

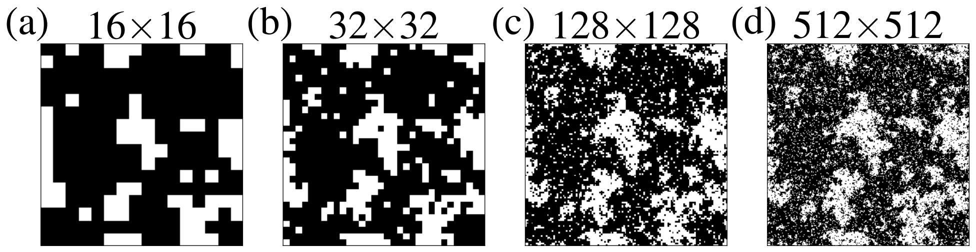


Figure 3.3: Critical configurations obtained using the weight extrapolation idea presented in Section 3.2.3. We show the original Monte Carlo configuration in (a) and the results after (b) one, (c) three and (d) five consecutive super-resolutions.

super-resolution could be used to access sizes that cannot be obtained by other means. We propose a simple method to do precisely this by exploiting the weight sharing property of convolutions. Namely, the size of the weight matrix W (and the bias vector $\mathbf{b}$) on a convolutional layer is independent of the input and output size.

In order to apply the convolution to a larger image, we only have to "slide" (Fig. 3.2b) the trained matrix W over a larger surface. The first upsampling layer does not contain any weights and can be trivially applied to any size. Therefore, using the weights of the trained $\mathbb{Z}_2^{L\times L} \to \mathbb{Z}_2^{2L\times 2L}$ network, we can define a new $\mathbb{Z}_2^{L'\times L'} \to \mathbb{Z}_2^{2L'\times 2L'}$ network that can be used to double any input size L'.

Accordingly, we can take the $2L \times 2L$ output of the first super-resolution as the input of a new network $\mathbb{Z}_2^{2L\times 2L} \to \mathbb{Z}_2^{4L\times 4L}$. Doing this repeatedly, we can create a chain of increasing sizes ($\mathbb{Z}_2^{16\times 16} \to \mathbb{Z}_2^{32\times 32} \to \mathbb{Z}_2^{64\times 64} \to \dots$). In this chain, only the smallest configurations are generated with MC, while the rest are result of successive super-resolutions with the same θ parameters. Figure 3.3a shows a configuration of the 2D Ising model at criticality, while Fig. 3.3 show the super-resolutions obtained from this configuration.

In summary, we use configurations $\boldsymbol{\sigma}_i \in \mathbb{Z}_2^{2L\times 2L}$ to generate decimated configurations $\mathcal{MR}(\boldsymbol{\sigma}_i) \in \mathbb{Z}_2^{L\times L}$. We then train a CNN network $\mathcal{SR}_\theta$ to invert this transformation. If the network is trained correctly, new configurations $\mathcal{SR}_\theta(\mathcal{MR}(\boldsymbol{\sigma}))$, should obey the Boltzmann distribution within reasonable error.

In the following Sections, we test this procedure by calculating observables in the 1D and 2D Ising models.

3.3 One-dimensional Ising model

In this Section, we show the results of the super-resolution scheme applied to the 1D Ising model. For this model, the Hamiltonian is self-similar under RG steps and the decimation is exactly solvable. This serves as a useful benchmark to test the validity of our super-resolution scheme before assessing the more interesting 2D case.

We begin with training the network with two lattice sizes, N and $2N$, before attempting extrapolation to larger sizes. We use a dataset consisting of temperatures ranging from $T = 0.01$ to $T = 3.5$ with $N = 32$ spins. At each temperature, we create training and testing sets consisting of $n = 10^4$ configurations generated via standard Monte Carlo. Instead of the majority rule, we use real-space block-spin decimation to obtain a $N = 16$ chain [104]. For each temperature, we implement a different network and optimize using the Adam method [59], with a batch size of 10^3. Instead of training for a specific amount of epochs, we cease training when the validation loss stops improving using early stopping.

The first step is to check that our network can correctly invert the decimation of a given sample. We directly evaluate the network's performance on the objective that it was trained on, namely the super-resolution of down-sampled (DS) configurations. DS configurations are obtained by applying the Kadanoff block-spin decimation transformation on the large MC samples [104]. DS configurations are then used as the network's input for training.

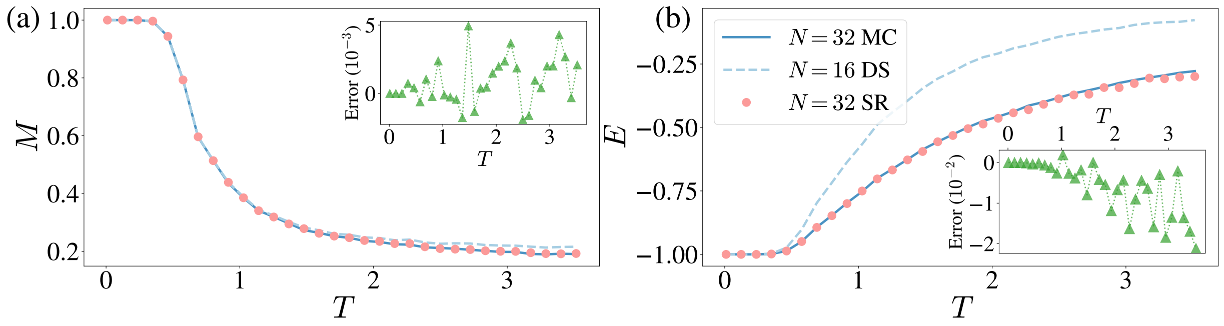


Figure 3.4: (a) Magnetization and (b) energy of the 1D Ising model. The dashed line corresponds to observables computed with the down-sampled (DS) configurations used as the network's input.

We find that the network achieves reasonable accuracy at each temperature when evaluated in the test set in Fig. 3.4. This shows that the network performs an approximate inverse of the block-spin decimation. We note that this inversion is demonstrated only at the level of thermodynamic observables and not the whole statistical ensemble. Mathematically, for any observable $\mathcal{O}$, the expectation values over the SR distribution and the original Boltzmann distribution are approximately the same: $\langle\mathcal{O}\rangle_{\mathcal{SR}_\theta(\mathcal{MR}(\boldsymbol{\sigma}))} \cong \langle\mathcal{O}\rangle_{\boldsymbol{\sigma}}$ where $\boldsymbol{\sigma}$ are the samples of the original (Boltzmann) distribution. In the remainder of this Section, we will focus on generating larger sizes than present in the training set.

So far, we have not discussed the temperature of configurations or the rescaling part of an RG step, but it will be essential to obtaining larger configurations. Conveniently, in 1D we can exactly calculate the marginalization of Eq. (3.2). The Hamiltonian is self-similar under the RG decimation with the rescaled couplings

$$\tilde{K} = f(K) = \frac{1}{2}\ln\cosh 2K\,. \tag{3.5}$$

Thus, under successive RG steps, the temperature T of a configuration flows towards infinity.

In effect, we train the network to take configurations $\mathbf{s}_i$ at couplings $\tilde{K}$ (obtained by applying $\mathcal{MR}$ to $\boldsymbol{\sigma}_i$ at K) and then super-resolve to a high-resolution configuration $\mathcal{SR}(\mathbf{s}_i)$ at K. If we use this configuration as the new input to the network, we can generate new, larger configurations indefinitely, following the method described in Section 3.2.3. After one super-resolving step, the new configurations produced by the network are $\mathcal{SR}(\mathcal{SR}(\mathbf{s}_i))$ at $f^{-1}(K) = f^{-1}(f^{-1}(\tilde{K}))$. Here we see that it is important to know how to apply the rescaling step in Eq. (3.5).

To validate our extrapolation proposal, we compare the magnetization and energy of super-resolved configurations with Monte Carlo results. Fig. 3.5 shows the results of the network after adjusting the temperature with Eq. (3.5). We stress that the network was trained on $\mathbb{Z}_2^{16} \to \mathbb{Z}_2^{32}$ data, yet predicts $N = 64$ accurately by extrapolation.

As another test of the super-resolving network, we repeat extrapolation up to $N = 512$ spins and calculate the two-point function, $G_N(j) = \langle\sigma_1\sigma_{1+j}\rangle$, for configurations from each generated size. In 1D, the exact value for this quantity from [115] is:

$$G_N(j;K) = \frac{\tanh^j K + \tanh^{N-j} K}{1 + \tanh^N K}\,. \tag{3.6}$$

In Fig. 3.6 we plot the two-point function for the different sizes with $j = N^{0.8}/5$. We note that the choice $j = N^{0.8}/5$ does not have a particular physical significance as it is possible to obtain similar accuracy for different choices of j.

Generally, errors are expected to increase with each super-resolution step. However, in the current case, the temperature flows towards zero under the inverse Eq. (3.5), so the extrapolation scheme remains stable even after multiple super-resolutions.

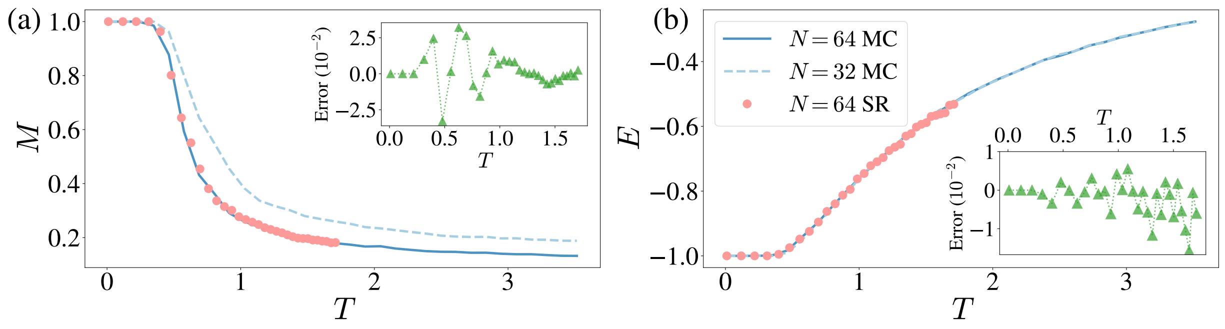


Figure 3.5: (a) Absolute magnetization and (b) energy per spin for the 1D Ising model. We denote Monte Carlo results at low ($N = 32$) and high ($N = 64$) resolution with MC. The super-resolution (SR) results were obtained by using the $N = 32$ MC data as input to an $\mathbb{Z}_2^{16} \to \mathbb{Z}_2^{32}$ network, and extrapolating new $N = 64$ configurations. SR temperatures are adjusted according to the inverse of Eq. (3.5). This shrinks the temperature range as the inverse RG transformation flows towards $T = 0$. Inset plots correspond to the error between SR predictions and MC results.

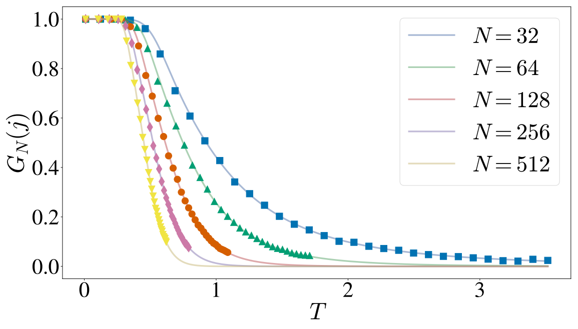


Figure 3.6: Two-point function of the 1D Ising model with $j = N^{0.8}/5$. Solid lines correspond to Eq. (3.6) and marked points to the super-resolution prediction. We use MC data for $N = 32$, while all other sizes are obtained from consecutive super-resolutions.

We have demonstrated numerically with two different methods that our super-resolution mapping can successfully capture thermodynamic quantities, as an approximate inverse RG transformation. The network parameter extrapolation is particularly effective in 1D where we know exactly how to rescale the temperature from Eq. (3.5).

3.4 Two-dimensional Ising model

Following the success of the 1D case, we proceed by training a 2D, $\mathbb{Z}_2^{8\times 8} \to \mathbb{Z}_2^{16\times 16}$ network using Monte Carlo generated datasets. The deterministic majority rule is now used instead of simple decimation. We again verify that the network works is able to invert the majority rule in Fig. 3.7.

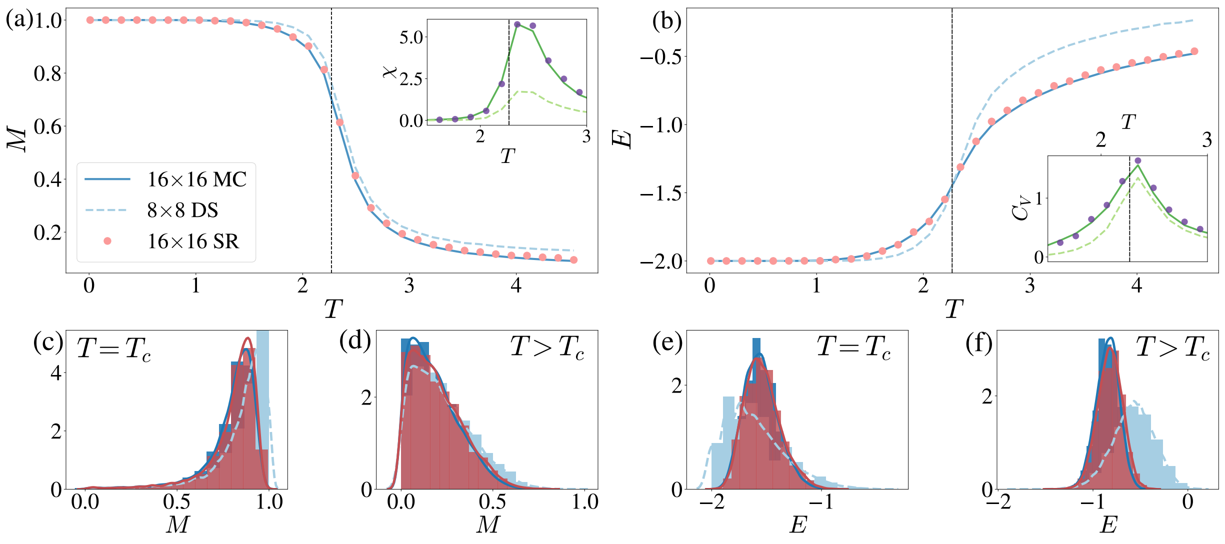


Figure 3.7: (a) Magnetization (with susceptibility) and (b) energy (with specific heat) of the 2D Ising model. MC denotes Monte Carlo results while SR is obtained by super-resolving the 8×8 downsampled (DS) configurations using the $\mathbb{Z}_2^{8\times 8} \to \mathbb{Z}_2^{16\times 16}$ network. Below: Probability distributions of magnetization and energy at $T = 2.2010 \simeq T_c$ for (c, e) and $T = 2.9313 > T_c$ for (d, f). The observables are binned into 15 bins to obtain these histograms. Colours follow the convention of the plots (a, b).

The main challenge in 2D is that the marginalization of Eq. (3.2) cannot be done analytically and hence we cannot simply rescale using Eq. (3.5). The 2D model is not self-similar under the block-spin RG transformation, as the Hamiltonian that corresponds to the decimated distribution contains interactions beyond nearest-neighbours [116].

To obtain a 2D analogy of Fig. 3.5, we approximate the temperature correction numerically for observables. In order to numerically find the transformation $f^{-1} : \tilde{K} \to K$ we compare observables calculated on an 8×8 MC configuration with those calculated on an 8×8 decimated one. We require that the corresponding curves collapse when the transformation is applied to the MC data. We note that this rescaling procedure has no direct physical interpretation since the 2D Ising Hamiltonian is not self-similar after a block spin RG transformation, and therefore temperature alone is not sufficient to describe the coupling space of the RG configuration. Thus, here we use this procedure only to demonstrate that our results are consistent with the inverse-RG nature of super-resolution.

3.4.1 Approximate rescaling

When extrapolating the network parameters, the new larger configurations generated are $\mathcal{SR}(\mathcal{SR}(\mathbf{s}_i))$ at $f^{-1}(K)$. The rescaling function f^{-1} was trivially found in 1D from the known analytical result. However, such a result does not exist in the 2D case. Here, we describe a method to approximate this rescaling function numerically.

Let $\mathbf{s}_i \in \mathbb{Z}_2^{8\times 8}$ denote the configurations obtained upon applying $\mathcal{MR}$ to the 16×16 Monte Carlo data $\boldsymbol{\sigma}_i$. For the purpose of this Section we also sample 8×8 Monte Carlo data denoted by $\boldsymbol{\tau}_i$. To find the rescaling, we compare the magnetization calculated on $\boldsymbol{\tau}_i$ with that from $\mathbf{s}_i$. We find the transformation $T \to \tilde{T}$ by requiring the corresponding $M(T)$ curves to collapse. An easy way to do so is shown in Fig. 3.8. Starting from a point at temperature $\tilde{T}$ in the $\boldsymbol{\tau}_i$ curve, one moves horizontally towards the $\mathbf{s}_i$ curve and the intersection defines T. By this construction, applying the rescaling to MC data, makes them collapse to downsampled (upon application of $\mathcal{MR}$) data of the same size. Therefore, this rescaling is equivalent to an approximation of f^{-1} used to rescale SR data points in Fig. 3.10.

We note that finding this rescaling is not related to the super-resolution procedure and it is not used in the critical exponent calculation, where we assume that we are at the fixed point of the $\mathcal{MR}$ transformation.

3.4.2 Importance of sampling

As mentioned in Section 3.2, the continuous output after the last sigmoid layer is interpreted as the probability that the spin in the corresponding site is up. Therefore,

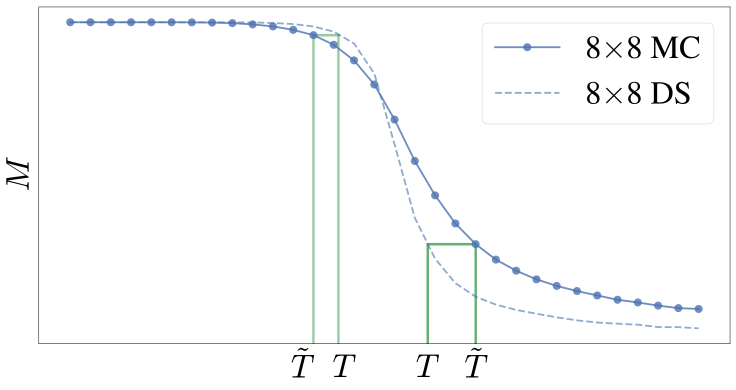


Figure 3.8: The method to find the $T \to \tilde{T}$ rescaling by approximately collapsing the MS and DS distributions of the same system size.

the super-resolved configuration is obtained by sampling this continuous output. The $\mathcal{F}_\theta : \mathbb{Z}_2^{L\times L} \to \mathbb{Z}_2^{2L\times 2L}$ mapping consists of the convolutional network and the sampling procedure.

This sampling procedure makes the $\mathcal{SR}$ mapping non-deterministic. We believe that sampling is crucial for the method to work. Here we give some numerical evidence to corroborate this statement.

The loss of information in the majority rule RG is associated with the different types of blocks that give the same decimated spin. For example, consider a block with 4-up/0-down spins and one with 3-up/1-down. Both would lead to an up spin in the decimated configuration. In order to capture the correct thermodynamics, the inverse RG procedure should give these different block types with the correct proportion at each temperature. To investigate whether this happens, we observe that the type of each block can be uniquely defined by the sum of spins contained in the block. In the $\mathbb{Z}_2 = \{0, 1\}$ convention, this sum is the number of up spins and goes from 0 to 4.

In Fig. 3.9 we give the number of appearances of each block sum in the original MC configurations and different interpretations of the network's output (rounding or sampling). The height of each bar is calculated by summing the appearances of each block sum over each configuration. At low temperatures, most configurations are fully polarized with the value 0 (all down) and 4 (all up). In this case, we do not have information loss during RG and there is no significant difference between rounding and sampling the output. At temperatures near and above criticality, non-fully-polarized blocks start to appear, increasing the appearance of intermediate sums (1 to 3). Rounding fails to capture the non-fully polarized blocks, making the use of sampling imperative. Even sampling cannot

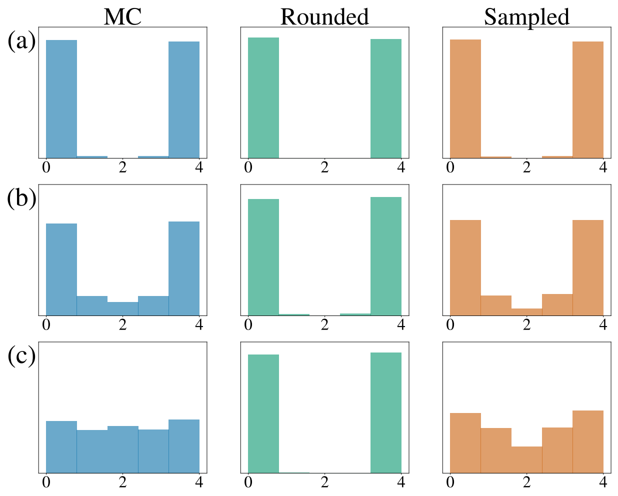


Figure 3.9: Histograms of the different 2×2 block sums at three different temperatures (a) $T = 1.4706$ (low), (b) $T = 2.2010$ (critical) and (c) $T = 2.9313$ (high). The first column corresponds to MC configurations, while the second and third to rounded and uniformly sampled network output respectively.

accurately capture the blocks with 2-up/2-down spins, indicating a possible systematic inaccuracy in our method that could be improved in further work.

3.4.3 Thermodynamic observables

We proceed by extrapolating the parameters of the trained $\mathbb{Z}_2^{8\times 8} \to \mathbb{Z}_2^{16\times 16}$ network and using it to super-resolve 16×16 MC configurations to 32×32. We present the results for magnetization and energy in Fig 3.10a,b. As in 1D, we see that the rescaling makes predicted SR observables match the MC results, indicating again that the network performs an approximate inversion of the RG transformation as desired.

We note that at high temperatures, the noise is largely random and difficult to learn. In contrast to 1D, we add a regularization term in the loss function, which compares the energy of the super-resolved configuration to that of the original one. This does not use any more information than already present in the training data, but results in better convergence of the network for high temperatures.

To corroborate our findings, we calculate the probability distributions of magnetization and energy in Fig. 3.10, for $T \simeq T_c$ and $T > T_c$. We see that super-resolution captures not

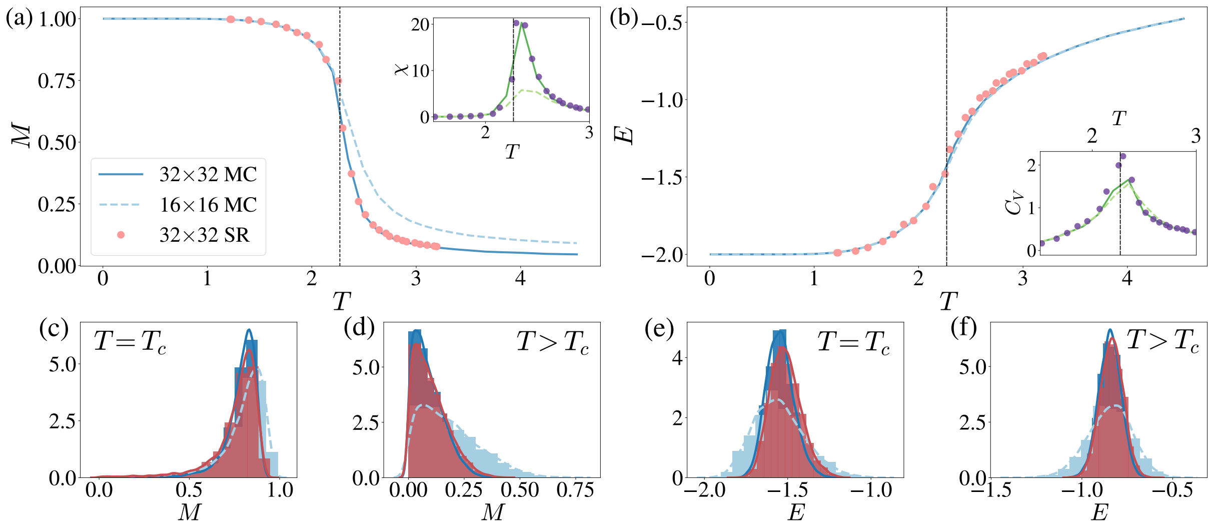


Figure 3.10: (a) Absolute magnetization with the susceptibility and (b) energy with specific heat for the 2D Ising model. MC denotes Monte Carlo results while SR is obtained by super-resolving the 16×16 MC configurations using the extrapolation of the $\mathbb{Z}_2^{8\times8} \to \mathbb{Z}_2^{16\times16}$ network. SR temperatures were rescaled using the numerical transformation, which shrinks temperature range towards criticality. (c), (d) Probability distributions of magnetization at $T = 2.2010 \simeq T_c$ for and $T = 2.9313 > T_c$, and (e), (f) the probability distributions for energy.

only the average values of magnetization and energy, but their entire probability distribution.

We expect that by increasing the extrapolation to larger sizes, any error in the original data or imperfections in the network will propagate. However, we suggest that for a single extrapolation, the network does remarkably well. In Fig. 3.11 we show the relative error in the energy after repeating the upsampling. We see here that the error grows with each successive upscaling as expected. This does not hold for all thermodynamic quantities, as the magnetization typically stays within 0.5% error at T_c for up to three super-resolutions. We propose that even multiple extrapolations may serve as a good starting point for large-scale simulations, e.g. possibly shortening equilibration time in a Monte Carlo procedure.

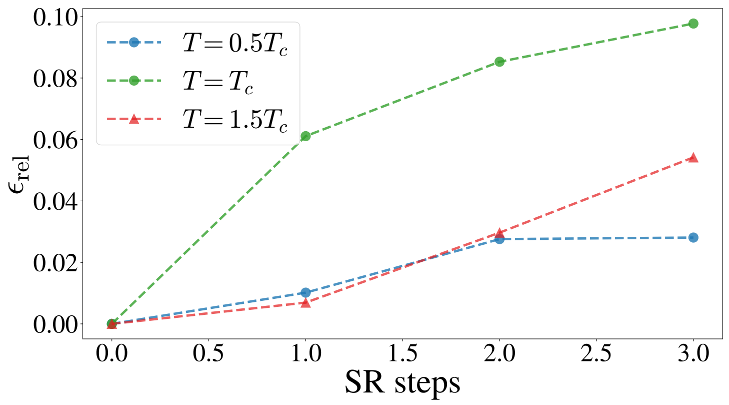


Figure 3.11: The relative error in the energy $\epsilon_{\mathrm{rel}} = |E_{\mathrm{MC}} - E_{\mathrm{SR}}|/E_{\mathrm{MC}}$ as a function of number of SR steps. At $T = T_c$, the algorithm is the most inaccurate, with around 6% error after one step.

3.4.4 Critical Exponents

An interesting application of super-resolution is the calculation of critical exponents. In principle, we can avoid rescaling if we focus on the fixed point of the RG transformation, where the Hamiltonian is self-similar. Here we crudely approximate (as per [113]) the fixed point with the nearest-neighbour Hamiltonian precisely at the critical temperature. We take 10^5 samples of 16×16 Monte Carlo configurations and repeatedly extrapolate to reach sizes up to 128×128. We stress that Monte Carlo simulation is required only on the smallest size (in our case 16×16) as all larger sizes are obtained by extrapolating.

We use the predicted configurations to calculate the critical exponents for the 2D Ising universality class using the finite-size scaling hypothesis [27]. According to this, exactly at criticality $\chi \propto L^{\gamma/\nu}$, where γ and ν are the susceptibility and correlation length critical exponents respectively. We can estimate the γ/ν ratio from the slope in a log-log (L, χ) plot in Fig. 3.12. Similarly, the two-point function vanishes algebraically $G(r) \sim 1/r^{d-2+\eta}$ at criticality, allowing us to estimate the anomalous dimension η from the slope of a log-log $(r, G(r))$ plot. The two-point function is calculated using two different values of the corresponding distance $r = L/4$ and $r = L/2$, leading to the two estimates η_1, η_2 for the anomalous dimension. Similarly, we can compute the magnetization exponent β. The exponents found through this method are presented in Table 3.1, where we see the remarkable agreement with analytical results. To get an estimation of our method's error in the critical exponents, we repeat the training and critical exponent calculation 60 times for sizes up to 128×128. We show the predicted values and percentage error in Table 3.1. The most important result is that we achieve less than 2% error from the theoretical value of the exponents in all cases.

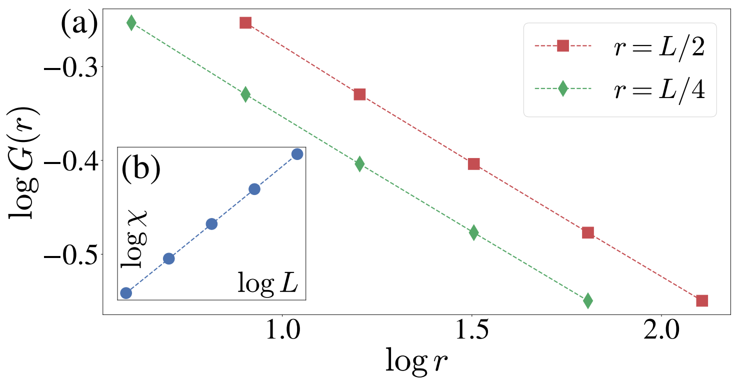


Figure 3.12: Scaling of the two-point function and susceptibility (inset) at criticality. The smallest size is calculated with Monte Carlo and the rest with repeated super-resolutions. Errors are typically around 10^{-3} and too small to show in this figure.

3.5 Discussion

We have investigated whether super-resolution techniques can be used to successfully increase the size of physical configurations sampled from the 1D and 2D Ising Hamiltonian. Inspired by recent applications of deep learning, we used a convolutional neural network for this task. We performed supervised training with a set of Monte Carlo configurations as output, and their corresponding RG-decimated counterparts as input. Therefore,

Exponent	Super-resolution	Error
β/ν	0.1234 ± 0.006	1.3%
γ/ν	1.7544 ± 0.01	0.25%
η_1	0.2460 ± 0.01	1.6%
η_2	0.2459 ± 0.01	1.6%

Table 3.1: Critical exponents of the 2D Ising universality class. We give the mean and standard error of 60 independent repetitions of training and critical exponent calculation from 16×16 to 128×128. The error is calculated in respect to exact values in the thermodynamic limit.

the network was essentially trained to double the size of configurations by performing a transformation approximately equivalent to an inverse RG step.

Despite the challenge in rigorously defining the inverse RG transformation due to the loss of information during the decimation, we found that our super-resolution scheme can accurately capture thermodynamic observables over a wide range of temperatures. We further proposed a method to extrapolate the trained weights and biases, and used them to access arbitrary lattice sizes larger than those used for training. We found that the extrapolation worked well for both 1D and 2D systems, and we were able to compute critical exponents in the 2D case which show agreement with analytical results to within 2%. To achieve this success, the method hinges on knowing the rescaling of the couplings $\tilde{K} = f(K)$ at each RG step. We expect that by modifying the network architecture to include an auxiliary parameter, one could learn this rescaling directly from the data, possibly with more accuracy than the approximations here.

Looking forward, these techniques may be beneficial to the large-scale simulation of complex physical systems. For example, our extrapolation may provide approximate initial configurations for further optimization procedures such as Monte Carlo updates. Decimation and super-resolution could be used to propose non-local updating procedures in models that suffer from long autocorrelation times, such as lattice quantum chromodynamics. Other interesting extensions of the current work could involve systems with disordered couplings, where more sophisticated RG techniques such as the energy based Ma-Dasgupta-Hu method, may be necessary [117, 118].

Ultimately, ideas analogous to the weight extrapolation might allow one to generate approximate configurations for lattice sizes that are inaccessible by other means. A quantum generalization could prove particularly useful as a way to generate approximate configurations that are beyond reach of current quantum Monte Carlo or tensor network methods.

4

Learning quantum states from measurements

With the current growth in quantum technology, it is increasingly important to develop reliable techniques that model the properties of quantum systems in order to diagnose and calibrate such devices. Due to the statistical nature of quantum mechanics, it is impossible to fully characterize a given quantum state with a single experimental measurement. Instead, thousands of repeated measurements must be performed on copies of the state. The ensemble statistics of these measurements can be used to gain insight into what state was prepared.

In this Chapter, we demonstrate the use of restricted Boltzmann machines (RBM) and some preliminary results with attention-based transformer models for this problem of quantum state estimation for measurements. Sections 4.1 through 4.5 consist of published work from [5]. Section 4.6 consists of unpublished notes.

4.1 Introduction

Advances in fabricating quantum technologies, as well as in reliable control of synthetic quantum matter, are leading to a new era of quantum hardware where highly pure quantum states are routinely prepared in laboratories. With the growing number of controlled quantum degrees of freedom (such as superconducting qubits, trapped ions, and ultracold atoms [119, 120, 121, 122]) reliable and scalable classical algorithms are required for the analysis and verification of experimentally prepared quantum states. Efficient algorithms can aid in extracting physical observables otherwise inaccessible from experimental measurements, as well as in identifying sources of noise to provide direct feedback for improving experimental hardware. However, traditional approaches for reconstructing unknown quantum states from a set of measurements, such as quantum state tomography, often suffer the exponential overhead that is typical of quantum many-body systems.

One of the fundamental aspects of quantum mechanics is that measuring a system changes it. This is a problem for experimental efforts that want to verify that their devices are preparing the correct quantum system. One solution is to perform repeated measurements of the final prepared system and infer the most likely state to explain the measurement outcomes. This process is known as quantum state tomography and generally requires a technique such as maximum likelihood estimation. In general this is an NP-hard problem, so approximate schemes are used.

For a random quantum state, the number of measurements needed for quantum state tomography is exponentially large in the number of degrees of freedom. However, physical interesting models often exhibit local interactions and various symmetries that can reduce the complexity of the system. For instance, in one-dimension, matrix product states (MPS) provide an efficient and tractable ansatz to represent states with low-entanglement [123, 124, 125, 126, 127]. For other systems however, new methods are needed to find efficient low-dimensional representations of a generic quantum many-body state.

Recently, an alternative path to quantum state reconstruction was put forward, based on machine learning (ML) techniques [128, 129, 130]. The most common approach relies on a generative model called a *restricted Boltzmann machine* (RBM) [131], a stochastic neural network with two layers of binary units. A visible layer $\boldsymbol{v}$ describes the physical degrees of freedom, while a hidden layer $\boldsymbol{h}$ is used to capture high-order correlations between the visible units. Given a set of neural network parameters $\boldsymbol{\theta}$, the RBM defines a probabilistic model described by the parametric distribution $p_{\boldsymbol{\theta}}(\boldsymbol{v})$. RBMs have been widely used in the ML community for the pre-training of deep neural networks [132], for compressing high-dimensional data into lower-dimensional representations [133], and more [16]. More recently, RBMs have been adopted by the physics community in the context of representing both classical and quantum many-body states [13, 99]. They are currently being investigated for their representational power [134, 135, 136], their relationship with tensor networks and the renormalization group [95, 107, 109, 108, 137], and in other contexts in quantum many-body physics [138, 139, 140].

In this Chapter, we demonstrate neural-network quantum state reconstruction of many-body wavefunctions can be used to reconstruct observables from projective measurement data. We can apply this idea to many types of data, including magnetic spin projections, orbital occupation number, polarization of photons, or the logical state of qubits. A properly trained neural network is an approximation of the unknown quantum state underlying the data. It can be used to calculate various physical observables of interest, including measurements that may not be possible in the original experiment.

This Chapter is organized as follows. In Section 4.2 we introduce restricted Boltzmann machines (RBMs) and how to train them. We proceed to use RBM to reconstruct positive-definite wavefunctions in Section 4.3. In Section 4.4, we consider the more general case of a complex-valued wavefunction. Finally, in Section 4.6 we explore using modern attention-based transformer architectures for tomography.

4.2 Restricted Boltzmann machines

The RBM is a physics-inspired model that had early success in machine learning, both in supervised and unsupervised learning [133]. They were originally introduced by Smolensky under the name *harmonium* [131] and were motivated by the Hopfield network [141]. They have also been called the inverse Ising problem in the statistical physics community [142].

RBMs are an undirected graphical model with binary nodes divided into two classes; *visible* and *hidden*. The idea is to use the hidden units to mediate interactions between visible units, while ultimately only caring about the distribution of visible units. Such techniques are widely used in physics, for instance the Hubbard-Stratonovich transformation mapping the Ising model to an effective ϕ^4-theory [143], or the Faddeev-Popov method of ghost fields in gauge theory [144]. In machine learning, this is referred to as a *latent variable model*. This similarity has deep connections to the Wilsonian and Kadanoff renormalization group (RG) [105, 95, 107].

The RBM consists of n_v visible units $\boldsymbol{v}$, which are binary valued, i.e. $v_i \in \{0, 1\}$. These represent an encoding of the data we are interested it. For example, this could be Ising spins or pixels from an image, or notes in a musical composition. The visible units are not inter-connected, however each visible unit is connected to every one of n_h hidden units $\boldsymbol{h}$ as per Fig. 4.1. Hidden units could be binary, multinomial, or real-valued, however for our applications we focus only on binary units. The term "restricted" refers to this limited connectivity between the units.

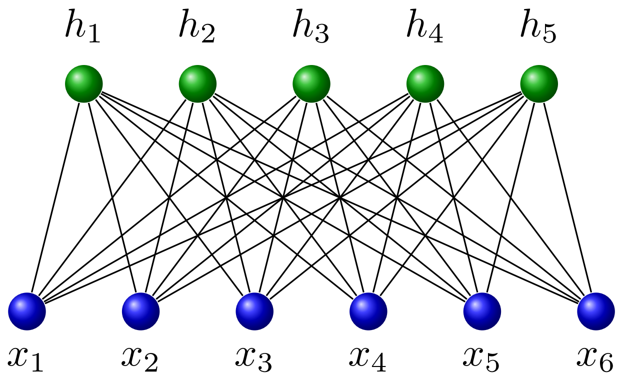


Figure 4.1: Restricted Boltzmann machine (RBM) with visible units $x_1, x_2 \ldots, x_6$, and hidden units $h_1, h_2, \ldots, h_5$.

Mathematically, for a given configuration of visible $\boldsymbol{v}$ and hidden $\boldsymbol{h}$ units the energy is

$$E_\theta(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i,j}^{n_v,n_h} v_i W_{ij} h_j - \sum_i^{n_v} a_i v_i - \sum_j^{n_h} b_j h_j \tag{4.1}$$

$$= -\boldsymbol{h}^T W \boldsymbol{v} - \boldsymbol{a}^T \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} \tag{4.2}$$

where W is the coupling matrix, also called the weight matrix, and a, b are called biases. The parameters a, b, W are all real-valued. From here it's clear why it is also known as the inverse Ising problem; instead of finding the ground state for fixed θ, we are interested in *learning* the best θ so that the ground state aligns with a different data set.

The joint probability for a visible-hidden configuration is given by the Boltzmann distribution

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z_\theta} \tag{4.3}$$

where $Z_\theta = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ is the partition function. Since Z_θ involves a sum over exponentially many terms, it is only feasible to compute for very small RBMs. Fortunately, to train the RBM we will not need Z_θ explicitly, only the ratio of probabilities.

The goal of the RBM is to model the probability distribution over visible units $p(\boldsymbol{v})$, so we need to trace out the hidden units:

$$p_\theta(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p_\theta(\boldsymbol{v}, \boldsymbol{h}) = \frac{e^{-\mathcal{E}(\boldsymbol{v})}}{Z_\theta}\,, \tag{4.4}$$

which defines $\mathcal{E}(\boldsymbol{v})$, the effective, or marginalized energy:

$$\mathcal{E}(\boldsymbol{v}) = -\log\left(\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}\right) \tag{4.5}$$

$$= -\sum_{j=1}^{N} a_j v_j - \sum_{i=1}^{N_h} \log\left(1 + e^{\sum_j W_{ij} v_j + b_i}\right) \tag{4.6}$$

$$= -\sum_{j=1}^{N} a_j v_j - \sum_{i=1}^{N_h} \ln\cosh\left(\sum_j W_{ij} v_j + b_i\right) + \text{const}\,. \tag{4.7}$$

$$\tag{4.8}$$

To gain some intuition behind the representation power of the RBM, we consider expanding the $\ln\cosh\left(\sum_j W_{ij} v_j + b_i\right)$ term as a Taylor series:

$$\ln\cosh x = \frac{x^2}{2} - \frac{x^4}{12} + \frac{x^6}{45} + O\left(x^8\right) \tag{4.9}$$

with $x = \sum_j W_{ij} v_j + b_i$. We see that the effective energy implicitly includes a term between a visible spin v_i and all other visible spin, $j \neq i$. In this way, hidden units allow correlations between visible units not possible otherwise. Moreover, each hidden unit contributes interactions at arbitrary high-order. For complex weights and biases, this term is a generalization of the Jastrow factor used commonly in variational Monte Carlo [12, 145]. This effective pairwise all-to-all interaction provides one reason for the vast representational power of the RBM.

4.2.1 Sampling

The bipartite structure of the RBM allows us to draw samples by using Gibbs updates as discussed for the 1d Ising model in Section 1.1.4. The probability $p(\boldsymbol{v}, \boldsymbol{h})$ factorizes into $p(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{h})p(\boldsymbol{v}|\boldsymbol{h})$ and $p(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{v})p(\boldsymbol{h}|\boldsymbol{v})$ from Bayes rule. Following the same steps as Section 1.1.4, we have that the conditionals factorize into the product of individual probabilities,

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{N_v} p\left(v_i|\boldsymbol{h}\right) \tag{4.10}$$

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j=1}^{N_h} p\left(h_j|\boldsymbol{v}\right) \tag{4.11}$$

where each individual probability is described by the logistic sigmoid function $\sigma(x) = (1+e^{-x})^{-1}$ as

$$p\left(v_i = 1|\boldsymbol{h}\right) = \sigma\left(a_i + \sum_{j=1}^{N_h} W_{ij} h_j\right) \tag{4.12}$$

$$p\left(h_j = 1|\boldsymbol{v}\right) = \sigma\left(b_j + \sum_{i=1}^{N_v} W_{ij} v_i\right) . \tag{4.13}$$

To sample, we first fix the visible units $\boldsymbol{v}$ (so that $p(\boldsymbol{v}) = 1$) and update the hidden units $\boldsymbol{h}$ according to $p(\boldsymbol{h}|\boldsymbol{v})$. Then we perform the converse by fixing the hidden units $\boldsymbol{h}$ and sampling the visible units according to $p(\boldsymbol{v}|\boldsymbol{h})$.

This is one reason it was essential to have a bipartite connectivity in the RBM. While Gibbs sampling can be fairly efficient, for complex-valued RBMs it fails and Metropolis or more advanced techniques such as parallel tempering are required [13].

As discussed in Section 1.1, near a phase transition sampling can suffer from critical slowing down. This problem is doubly bad for the RBM since sampling can be slow, and simultaneous, the parameters could also be stuck in a local energy minima.

4.2.2 Training

The goal of generative modelling is to approximate data drawn from an unknown distribution $p(\mathbf{x})$ with a machine learning model with adjustable parameters $p_{\boldsymbol{\theta}}(\mathbf{x})$. One can measure the difference between the two distributions with the Kullback-Leibler (KL) divergence, also known as the relative entropy,

$$\mathbb{KL}(p\|p_{\boldsymbol{\theta}}) = \sum_{\mathbf{x}} p(\mathbf{x}) \ln p(\mathbf{x}) - \sum_{\mathbf{x}} p(\mathbf{x}) \ln p_{\boldsymbol{\theta}}(\mathbf{x}) \tag{4.14}$$

where $\mathbf{x}$ is a state in the probability space. The first term is just the Shannon entropy of the data set and the second term is the cross entropy. The KL divergence is always non-negative and is only zero if and only if $p(\mathbf{x}) = q(\mathbf{x})$. Moreover, it is not symmetric, $\mathbb{KL}(p\|q) \neq \mathbb{KL}(q\|p)$ so it is not a true metric between distributions.

Consider the case of a finite set of N independent samples $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N)$ drawn from the data distribution

$$p_{\text{data}}(\mathbf{x}) = N^{-1} \sum_{n=1}^{N} \delta(\mathbf{x} - \mathbf{x}_n) \tag{4.15}$$

The cross entropy term then simplifies to the average log-likelihood $\mathcal{L}_\theta$:

$$\mathcal{L} = N^{-1} \sum_{\mathbf{x}_n \in \mathcal{D}} \ln p_{\boldsymbol{\theta}}(\mathbf{x}_n)\,. \tag{4.16}$$

Minimizing the KL divergence is thus equivalent to minimizing the negative log-likelihood. This is known as maximum likelihood estimation.

We proceed to maximizing the log-likelihood via gradient ascent. The gradient of $\mathcal{L}_\theta$ with respect to the parameters $\boldsymbol{\theta}$ is

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta \left(N^{-1} \sum_{\mathbf{x}_n \in \mathcal{D}} \ln p_{\boldsymbol{\theta}}(\mathbf{x}_n) \right) \\
&= \nabla_\theta \left(N^{-1} \sum_{\mathbf{x}_n \in \mathcal{D}} \mathcal{E}_\theta(\mathbf{x}_n) - \ln Z_\theta \right) \\
&= N^{-1} \sum_{\mathbf{x}_n \in \mathcal{D}} \nabla_\theta \mathcal{E}_\theta(\mathbf{x}_n) - Z_\theta^{-1} \sum_{\mathbf{x}} \nabla_\theta p_{\boldsymbol{\theta}}(\mathbf{x}_n) \\
&= N^{-1} \sum_{\mathbf{x}_n \in \mathcal{D}} \nabla_\theta \mathcal{E}_\theta(\mathbf{x}_n) - \sum_{\mathbf{x}} p_{\boldsymbol{\theta}}(\mathbf{x}) \nabla_\theta \mathcal{E}_\theta(\mathbf{x}) \\
&= \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}_n) \rangle_{\mathbf{x} \in \mathcal{D}, \boldsymbol{h} \sim p_{\boldsymbol{\theta}}(\boldsymbol{h}|\mathbf{x}_n)} - \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}) \rangle_{(\boldsymbol{v},\boldsymbol{h}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{v},\boldsymbol{h})}
\end{aligned} \tag{4.17}$$

The first term is called the positive phase and involves averaging only over the sample data $\mathcal{D}$ and sampling $\boldsymbol{h} \sim p_{\boldsymbol{\theta}}(\boldsymbol{h}|\mathbf{x}_n)$ for each data point $\mathbf{x}_n \in \mathcal{D}$. The second term poses more problems since it involves an intractable sum over all possible states $\mathbf{x}$. The two phases compete to adjust the weights. In the positive phase, the model gravitates towards configurations in the data set. It assigns higher probability (and hence lower energy) to states that are more frequent in the data set. Conversely, the negative phase generates configurations from the RBM which favours exploring the state space. This allows generalization and a type of bias towards distributions $p_{\boldsymbol{\theta}}$ that are well suited to the Ising type problem. There can refereed to as learning and unlearning phases where the RBM first matches the training data in the positive (learning) phase, then expands to new configurations in the unlearning (negative) phase.

The positive phase is straightforward to compute. For each sample $\mathbf{x}_n$ in $\mathcal{D}$, we fix the visible units of the RBM to $\mathbf{x}_n$ and sample the hidden units (once) using block Gibbs from Eq. 4.10. This is generally fast since it involves only one computation per sample hence is $O(N)$.

The negative phase presents a considerable challenge. The standard approach is to draw samples $x \sim p_{\boldsymbol{\theta}}(\mathbf{x})$ using Markov chain Monte Carlo (MCMC). The advantage of MCMC is that it is generally applicable to many forms of $p_{\boldsymbol{\theta}}$. However, it also introduces three main challenges. Firstly, equilibrating the Markov chain at each parameter can be prohibitively slow. Moreover, without a global update, Gibbs sampling can get stuck in local minima. This mode-mixing problem can be partial combated with more advanced sampling techniques such as parallel tempering or fast-weights RBMs [146]. Lastly, the most serious problem is that variance of the gradient can be unmanageable. As the difference between two expectation values, the gradient Eq. 4.17 suffers if either term has high variance.

A solution to the variance problem was proposed by Hinton in 2002 [132]. The idea is we can bias our MCMC towards the data distribution to reduce the variance between the phases and quicken the equilibration time. Instead of sampling from $p_{\boldsymbol{\theta}}$ with MCMC, we first start with the visible units fixed to a sample $\mathbf{x}_n$. This reduces the tendency of the chain to drift away from the data samples, hence reducing the overall variance. We now use block Gibbs to sample the hidden units $\boldsymbol{h}$ using $p(\boldsymbol{h}|\mathbf{x}_n)$, and then fix the hidden units and sampled the visible units from $p(\mathbf{x}|\boldsymbol{h})$. At this point the visible units $\mathbf{x}$ might not be samples from the data set anymore. We introduce the notation $p_{\boldsymbol{\theta}}^k$, for k repetitions of this procedure. Repeating this procedure k-times results in a MCMC chain $p_{\boldsymbol{\theta}}^k$ that still converges to $p_{\boldsymbol{\theta}}$ in the limit $k \to \infty$. In this notation, the data distribution $(\mathbf{x} \in \mathcal{D}, \boldsymbol{h} \sim p_{\boldsymbol{\theta}}(\boldsymbol{h}|\mathbf{x}_n))$ is called p^0, and the equilibration RBM distribution is $p^\infty = p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})$.

This method is called *contrastive divergence* because it can be interpreted as minimizing the difference between two KL divergences:

$$\mathrm{CD}_k = \mathbb{KL}(p_{\text{data}}|p_{\boldsymbol{\theta}}) - \mathbb{KL}(p_{\boldsymbol{\theta}}^k|p_{\boldsymbol{\theta}})\,. \tag{4.18}$$

Since we are guaranteed that each k step is closer to the true distribution (i.e. $\mathbb{KL}(p_{\boldsymbol{\theta}}^{k+1}|p_{\boldsymbol{\theta}}) \leq$

$\mathbb{KL}(p^k_{\boldsymbol{\theta}}|p_{\boldsymbol{\theta}})$ with equality only if at equilibrium), we know that the contrastive divergence CD_k is non-negative.

The gradient of the contrastive divergence CD_k is

$$\nabla_\theta \mathrm{CD}_k = \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}_n) \rangle_{p^0} - \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}) \rangle_{p^k} - \sum_{\mathbf{x}} \nabla_\theta p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p^k_{\boldsymbol{\theta}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \tag{4.19}$$

The third term is problematic since it involved another exponential sum. It represents the effect that a change in θ results in a small change in $\mathbb{KL}(p^k_{\boldsymbol{\theta}}|p_{\boldsymbol{\theta}})$. For small changes in parameters, this term should remain small [147, 148]. This term can also be used to construct examples where CD_k fails to converge or converges to a different fixed point than maximum likelihood [149, 150, 151]. To proceed, we must neglect this term so that the gradient follows

$$\nabla_\theta \mathrm{CD}_k \approx \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}_n) \rangle_{p^0} - \langle \nabla_\theta \mathcal{E}_\theta(\mathbf{x}) \rangle_{p^k} \ .$$

This also means that contrastive divergence is not technically the gradient of any loss function [152]. In practice, $k = 1$ is sufficient for many problems [40], however it is biased towards the data distribution. Some papers suggest using CD_k to find an approximate solution, then tuning the model with maximum likelihood. One could also increase k near the end of training to reduce the bias from this approximation.

The energy gradients $\nabla_\theta \mathcal{E}_\theta(\mathbf{v})$ for the biases a_i, b_i, and the weights W_{ij} work out to be

$$\nabla_{a_i} \mathcal{E}_\theta(\boldsymbol{v}) = v_i \tag{4.20}$$
$$\nabla_{b_j} \mathcal{E}_\theta(\boldsymbol{v}) = p(h_j = 1|\boldsymbol{v}) \tag{4.21}$$
$$\nabla_{W_{ij}} \mathcal{E}_\theta(\boldsymbol{v}) = v_i \, p(h_j = 1|\boldsymbol{v}) \tag{4.22}$$

where p is given by Eq. 4.12.

The parameters are optimized with gradient descent

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathrm{CD}_k \tag{4.23}$$

for a learning rate η. More advance optimizers such as Adam, Adagrad, or natural gradient descent could be used instead[1].

[1]See Chapter 1, Section 1.2.3 for more information.

4.3 Positive wavefunctions with RBMs

As an example of using RBM for wavefunction reconstruction, we first consider the case of a positive-definite wavefunction. Consider a state $|\psi\rangle$ in Hilbert space written in terms of a complete basis $|\mathbf{x}\rangle$,

$$|\psi\rangle = \sum_{\mathbf{x}} \psi(\mathbf{x}) \, |\mathbf{x}\rangle \tag{4.24}$$

where the coefficients $\psi(\mathbf{x})$ are non-negative. The probability that the state collapses to $|\mathbf{x}\rangle$ after a measurement is related to $\psi(\mathbf{x})$ via the Born rule $p(\mathbf{x}) = |\psi(\mathbf{x})|^2$. With repeated projective measurements, an experimentalist can build a dataset $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots)$. Alternatively, samples $\mathbf{x}_n$ could be produced with simulations with quantum Monte Carlo, DMRG, or other techniques. In this Chapter, we generate the data for the TFIM with Monte Carlo described in Section 3.1.

Generally this dataset can provide value information about the system being studied. For instance, diagonal observables can be computed directly. However, off-diagonal observables or more complex quantities such as the Rényi entropy, cannot be calculated from the diagonal measurements alone. For instance, computing the Rényi entropy would require double the number of qubits. Instead, we use the RBM as a compressed representation of the quantum state similar to tensor networks or matrix product states [123]. After the RBM is trained on the dataset, it can be sampled to produce new observables.

The restriction to non-negative states makes the interpretation of the wavefunction as an RBM clear. The assumption of positivity is well-founded for many physically interesting systems, such as the ground states of Hamiltonians that do not encounter the sign problem [153]. In this Chapter, we focus on the TFIM in one-dimension which has non-negative elements in both the σ^x, or σ^z basis.

The RBM is trained to approximate the data distribution $p_{\text{data}}(\mathbf{x})$ with a generative model $p_{\boldsymbol{\theta}}(\mathbf{x})$. Using contrastive divergence (CD) [132] and gradient descent we train the RBM to discover an optimal set of parameters $\boldsymbol{\theta}$. Upon successful training, we obtain an approximate representation of the target quantum state,

$$\psi_{\boldsymbol{\theta}}(\mathbf{x}) \equiv \sqrt{p_{\boldsymbol{\theta}}(\mathbf{x})} \simeq \psi(\mathbf{x}) \, . \tag{4.25}$$

The quality of the representation can be measured with the KL divergence for small systems, or alternatively, with the fidelity $f(\psi, \psi_\theta) = |\langle\psi|\psi_\theta\rangle|^2$. For lager systems both the KL divergence and the fidelity becomes intractable; so we resort to comparing observables such as the energy, magnetization or entanglement entropy.

Consider a data set $\mathcal{D}$ composed of projective measurements generated by sampling the distribution the ground state of the TFIM model. We use Monte Carlo sampling from Appendix A, to generate configurations and their exact amplitudes. We consider $N_S = 10^5$ independent spin projections in the reference basis $\mathbf{x} = \boldsymbol{\sigma}^z$ and focus on the critical point $h/J = 1$.

In Fig. 4.2, we show that during training, the RBM successfully improves the fidelity and minimized the KL divergence for $N = 10$ qubits at the critical point $h = 1$ of the TFIM. The critical point is the most difficult to learn (in the thermodynamic limit) since correlations decay algebraically.

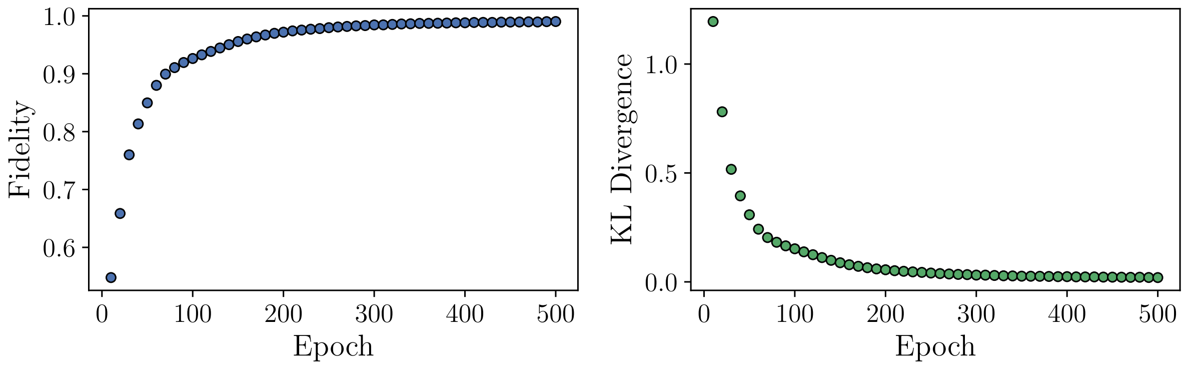


Figure 4.2: The fidelity (left) and the KL divergence (right) during training for the reconstruction of the ground state of the one-dimensional TFIM for $N = 10$ spins at the critical point $h/J = 1$.

Reconstruction of physical observables

In this Section, we discuss how to calculate the average value of a generic physical observable $\hat{\mathcal{O}}$ from a trained RBM. We start with the case of observables that are diagonal in the reference basis where the RBM was trained. We then discuss the more general cases of off-diagonal observables and entanglement entropy.

Diagonal observables

We begin by considering an observable with only diagonal matrix elements, $\langle \mathbf{x}| \hat{\mathcal{O}} |\mathbf{x}'\rangle = \mathcal{O}_{\mathbf{x}}\delta_{\mathbf{x}\mathbf{x}'}$ where for convenience we denote the computational basis σ^z as $\mathbf{x}$ unless otherwise stated. The expectation value of $\hat{\mathcal{O}}$ is given by

$$\langle \hat{\mathcal{O}} \rangle = \frac{1}{\sum_{\mathbf{x}} |\psi_{\boldsymbol{\theta}}(\mathbf{x})|^2} \sum_{\mathbf{x}} \mathcal{O}_{\mathbf{x}} |\psi_{\boldsymbol{\theta}}(\mathbf{x})|^2 \, . \tag{4.26}$$

The expectation value can be approximated by a Monte Carlo estimator,

$$\langle \hat{\mathcal{O}} \rangle \approx \frac{1}{N_{\mathrm{MC}}} \sum_{k=1}^{N_{\mathrm{MC}}} \mathcal{O}_{\mathbf{x}_k} \, , \tag{4.27}$$

where the spin configurations $\mathbf{x}_k$ are sampled from the RBM distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$. This process is particularly efficient given the bipartite structure of the network which allows the use of block Gibbs sampling.

A simple example for the TFIM is the average longitudinal magnetization per spin, $\langle \hat{\sigma}^z \rangle = \sum_j \langle \hat{\sigma}^z_j \rangle / N$, which can be calculated directly on the spin configuration sampled by the RBM (i.e., the state of the visible layer) as shown in Fig. 4.3. Other diagonal observables include the spin-spin correlation function $\langle \sigma^z_0 \sigma^z_n \rangle$, or in the case of fermions, the occupation number $n_i = \left\langle a_i^\dagger a_i \right\rangle$.

Off-diagonal observables

We turn now to the case of off-diagonal observables, where the expectation value assumes the following form

$$\langle \hat{\mathcal{O}} \rangle = \frac{1}{\sum_{\mathbf{x}} |\psi_{\boldsymbol{\theta}}(\mathbf{x})|^2} \sum_{\boldsymbol{\sigma}\boldsymbol{\sigma}'} \psi^*_{\boldsymbol{\theta}}(\mathbf{x}) \psi_{\boldsymbol{\theta}}(\mathbf{x}') \mathcal{O}_{\boldsymbol{\sigma}\boldsymbol{\sigma}'} \, . \tag{4.28}$$

This expression can once again be approximated with a Monte Carlo estimator

$$\langle \hat{\mathcal{O}} \rangle \approx \frac{1}{N_{\mathrm{MC}}} \sum_{k=1}^{N_{\mathrm{MC}}} \mathcal{O}^{[L]}_{\mathbf{x}_k} \tag{4.29}$$

of the so-called *local estimator* of the observable:

$$\mathcal{O}^{[L]}_{\mathbf{x}_k} = \sum_{\mathbf{x}'} \frac{\psi_{\boldsymbol{\theta}}(\mathbf{x}')}{\psi_{\boldsymbol{\theta}}(\mathbf{x}_k)} \mathcal{O}_{\mathbf{x}_k \mathbf{x}'} \, . \tag{4.30}$$

As long as the matrix representation $\mathcal{O}_{\boldsymbol{\sigma}\boldsymbol{\sigma}'}$ is sufficiently sparse in the reference basis, the summation can be evaluated efficiently since the normalization constant for ψ cancels out.

As an example, we consider the specific case of the transverse magnetization for the j-th spin, $\langle\hat{\sigma}^x_j\rangle$, with matrix elements

$$\langle\mathbf{x}|\hat{\sigma}^x_j|\mathbf{x}'\rangle = \delta_{\sigma'_j,1-\sigma_j}\prod_{i\neq j}\delta_{\sigma'_i,\sigma_j}\ . \tag{4.31}$$

Therefore, the expectation values reduces to the Monte Carlo average of the local observable

$$(\sigma^x_j)^{[L]} = \frac{\psi_\theta(\sigma_1,\dots,1-\sigma_j,\dots,\sigma_N)}{\psi_{\boldsymbol{\theta}}(\sigma_1,\dots,\sigma_j,\dots,\sigma_N)}\ . \tag{4.32}$$

The transverse magnetization is then computed by sampling $p_{\boldsymbol{\theta}}$ using block Gibbs sampling.

The reconstruction of two magnetic observables for the TFIM is shown in Fig. 4.3, where a different RBM was trained for each value of the transverse field h. In the left plot, we show the average longitudinal magnetization per site, which can be calculated directly from the configurations sampled by the RBM. In the right plot, we show the off-diagonal observable of transverse magnetization. For both cases, the RBM successfully discovers an optimal set of parameters $\boldsymbol{\theta}$ that accurately approximate the ground-state wavefunction underlying the data.

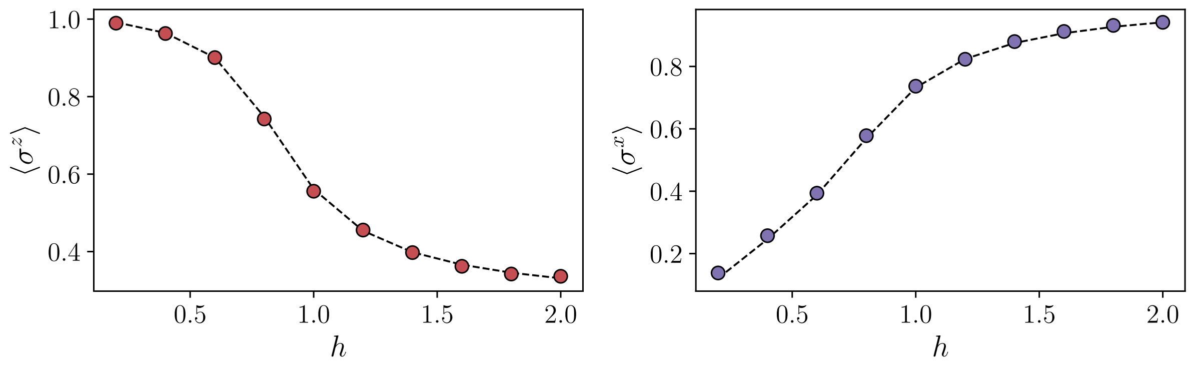


Figure 4.3: Reconstruction of the magnetic observables for the TFIM chain with $N = 10$ spins. We show the average longitudinal magnetization $\langle\sigma\rangle^z$ (left) and transverse magnetization $\langle\sigma\rangle^x$ (right) per site obtained by sampling from a trained RBM. The dashed line denotes the results from exact diagonalization.

Entanglement entropy

A quantity of significant interest in quantum many-body systems is the degree of entanglement between a subregion A and its complement $\bar{A}$. Numerically, measurement of bipartite entanglement entropy is commonly accessed through the computation of the second Rényi entropy $S_2 = -\ln \mathrm{Tr}(\rho_A^2)$ [154]. When one has access to a pure state wavefunction $\psi_{\boldsymbol{\theta}}(\mathbf{x})$, Rényi entropy can be calculated as an expectation value of the "Swap" operator,

$$S_2 = -\ln \left\langle \widehat{\mathrm{Swap}}_A \right\rangle . \tag{4.33}$$

It is essentially an off-diagonal observable that acts on an extended product space consisting of two independent copies of the wavefunction, $\psi_{\boldsymbol{\theta}}(\mathbf{x}) \otimes \psi_{\boldsymbol{\theta}}(\mathbf{x})$, referred to as "replicas". As the name suggests, the action of the Swap operator is to swap the spin configurations in region A between the replicas,

$$\widehat{\mathrm{Swap}}_A |\mathbf{x}_A, \mathbf{x}_{\bar{A}}\rangle_1 \otimes |\mathbf{x}'_A, \mathbf{x}'_{\bar{A}}\rangle_2 = |\mathbf{x}'_A, \mathbf{x}_{\bar{A}}\rangle_1 \otimes |\mathbf{x}_A, \mathbf{x}'_{\bar{A}}\rangle_2. \tag{4.34}$$

Here the subscript of the ket indicates the replica index, while the two labels inside a ket, such as $\mathbf{x}_A, \mathbf{x}_{\bar{A}}$, describe the spins configurations within the subregion and its complement.

Using this observable, we can estimate the second Rényi entropy of the region containing the first half of the chain using Eq. 4.33 as shown in Fig. 4.4. As was the case with the magnetization observables, the trained RBM gives a good approximation to the second Rényi entropy for different subregion A sizes. Being a basis-independent observable, this constitutes a useful test on the ability of the RBM to capture the full wavefunction from the information contained in a single-basis dataset for TFIM.

4.4 Complex wavefunctions

For positive wavefunctions, the probability distribution underlying the outcomes of projective measurements in the reference basis contains all possible information about the unknown quantum state. However, in the more general case of a wavefunction with a non-trivial sign or phase structure, this is not the case. In this section, we consider a target quantum state where the wavefunction coefficients in the reference basis can be complex-valued, $\psi(\mathbf{x}) = |\phi(\mathbf{x})| e^{i\Phi(\mathbf{x})}$. We continue to choose the reference basis as $\mathbf{x} = (\sigma_1^z, \sigma_2^z, \ldots)$. We first need to generalize the RBM representation of the quantum state to capture a generic complex wavefunction. To this end, we introduce an additional RBM

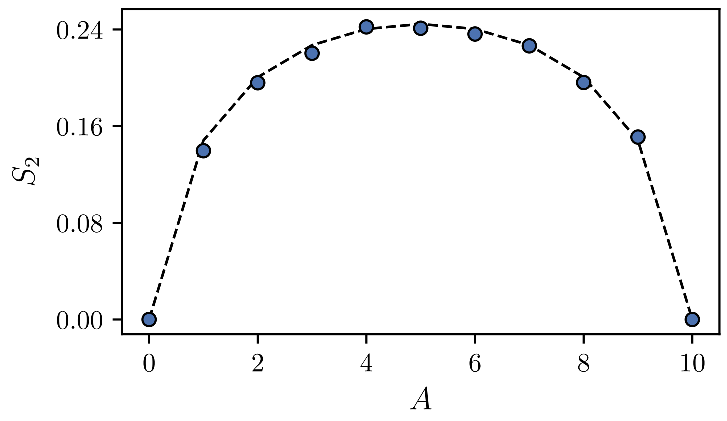


Figure 4.4: The second Rényi entropy for the TFIM chain with $N = 10$ spins. The number of sites in the entangled bipartition A is indicated by the horizontal axis. Markers indicate values obtained through the "Swap" operator applied to the samples from a trained RBM. The dashed line denotes the result from exact diagonalization.

with marginalized distribution $p_{\boldsymbol{\phi}}(\mathbf{x})$ parameterized by a new set of network weights and biases $\boldsymbol{\phi}$. We use this to define the quantum state as:

$$\psi_{\boldsymbol{\theta\phi}}(\mathbf{x}) = \sqrt{p_{\boldsymbol{\theta}}(\mathbf{x})}\, e^{iq_{\boldsymbol{\phi}}(\mathbf{x})/2} \tag{4.35}$$

where $q_{\boldsymbol{\phi}}(\mathbf{x}) = \log p_{\boldsymbol{\phi}}(\mathbf{x})$ [101]. In this case, the reconstruction requires a different type of measurement setting. It is easy to see that projective measurements in the reference basis do not convey any information on the phases $\theta(\mathbf{x})$, since $p(\mathbf{x}) = |\psi(\mathbf{x})|^2 = \Phi^2(\mathbf{x})$.

The general strategy to learn a phase structure is to apply a unitary transformation $\boldsymbol{\mathcal{U}}$ to the state $|\psi\rangle$ before the measurements, such that the resulting measurement distribution $P'(\mathbf{x}) = |\psi'(\mathbf{x})|^2$ of the rotated state $\psi'(\mathbf{x}) = \langle\mathbf{x}|\, \boldsymbol{\mathcal{U}}\, |\psi\rangle$ contains fingerprints of the phases $\Phi(\mathbf{x})$ (Fig. 4.5). In general, different rotations must be independently applied to gain full information on the phase structure. We make the assumption of a tensor product structure of the rotations, $\boldsymbol{\mathcal{U}} = \bigotimes_{j=1}^{N} \hat{\mathcal{U}}_j$. This is equivalent to a local change of basis from $|\mathbf{x}\rangle$ to $\{|\mathbf{x}^{\boldsymbol{b}}\rangle = |\sigma_1^{b_1}, \dots, \sigma_N^{b_N}\rangle\}$, where the vector $\boldsymbol{b}$ identifies the local basis b_j for each site j. The target wavefunction in the new basis is given by

$$\begin{aligned} \psi(\mathbf{x}^{\boldsymbol{b}}) = \langle \mathbf{x}^{\boldsymbol{b}}|\psi\rangle &= \sum_{\mathbf{x}} \langle \mathbf{x}^{\boldsymbol{b}}|\mathbf{x}\rangle\langle\mathbf{x}|\psi\rangle \\ &= \sum_{\mathbf{x}} \boldsymbol{\mathcal{U}}(\mathbf{x}^{\boldsymbol{b}}, \mathbf{x})\psi(\mathbf{x})\,, \end{aligned} \tag{4.36}$$

and the resulting measurement distribution is

$$p_{\boldsymbol{b}}(\mathbf{x}^{\boldsymbol{b}}) = \left| \sum_{\mathbf{x}} \boldsymbol{\mathcal{U}}(\mathbf{x}^{\boldsymbol{b}}, \mathbf{x}) \psi(\mathbf{x}) \right|^2 . \tag{4.37}$$

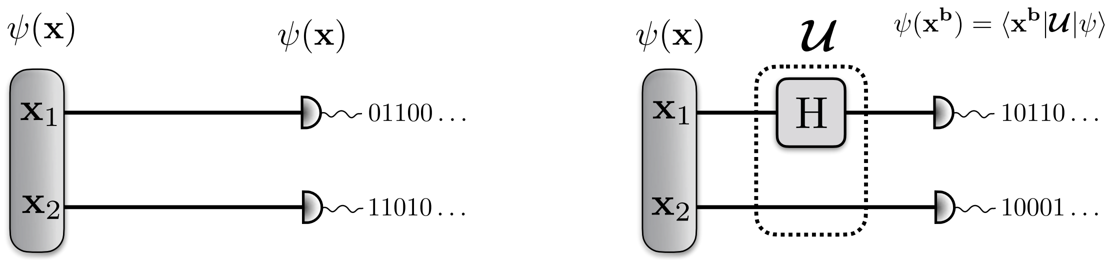


Figure 4.5: Unitary rotations for two qubits. (left) Measurements on the reference basis. (right) Measurement in the rotated basis. The unitary rotation (the Hadamard gate on qubit σ_0) is applied after state preparation and before the projective measurement.

To clarify the procedure, let us consider the simple example of a quantum state of two qubits:

$$|\psi\rangle = \sum_{\sigma_0,\sigma_1} \Phi_{\sigma_0\sigma_1} e^{i\theta_{\sigma_0\sigma_1}} |\sigma_0\sigma_1\rangle , \tag{4.38}$$

and rotation $\boldsymbol{\mathcal{U}} = \hat{\mathrm{H}}_0 \otimes \hat{\mathcal{I}}_1$, where $\hat{\mathcal{I}}$ is the identity operator and

$$\hat{\mathrm{H}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.39}$$

is called the *Hadamard gate*. This transformation is equivalent to rotating the qubit σ_0 from the reference σ_0^z basis to the σ_0^x basis. A straightforward calculation leads to the following probability distribution of the projective measurement in the new basis $|\sigma_0^x, \sigma_1\rangle$:

$$p_{\boldsymbol{b}}(\sigma_0^x, \sigma_1) = \frac{\Phi_{0\sigma_1}^2 + \Phi_{1\sigma_1}^2}{4} + \frac{1 - 2\sigma_0^x}{2} \Phi_{0\sigma_1} \Phi_{1\sigma_1} \cos(\Delta\theta) , \tag{4.40}$$

where $\Delta\theta = \theta_{0\sigma_1} - \theta_{1\sigma_1}$. Therefore, the statistics collected by measuring in this basis implicitly contains partial information on the phases. To obtain the full phases structure, additional transformations are required, one example being the rotation from the reference basis to the σ_j^y local basis, realized by the elementary gate

$$\hat{\mathrm{K}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ 1 & i \end{bmatrix} . \tag{4.41}$$

We now proceed to reconstruct a complex-valued wavefunction. For simplicity, we restrict ourselves to two qubits and consider the general case of a quantum state with random amplitudes $\Phi_{\sigma_0\sigma_1}$ and random phases $\theta_{\sigma_0\sigma_1}$. In contrast with the positive case, we now have measurements performed in different bases. Therefore, the training data consists of an array of qubits projections $(\sigma_0^{b_0}, \sigma_1^{b_1})$, together with the corresponding bases (b_0, b_1) where the measurement was taken. We also need the various elementary unitary rotations that need to be applied to the RBM state during the training. For this example, we generated measurements in the following bases:

$$(b_0, b_1) = (\mathrm{z}, \mathrm{z})\ ,\ (\mathrm{x}, \mathrm{z})\ ,\ (\mathrm{z}, \mathrm{x})\ ,\ (\mathrm{y}, \mathrm{z})\ ,\ (\mathrm{z}, \mathrm{y}) \tag{4.42}$$

Finally, before the training, we initialize the set of unitary rotations In the case of the provided 2-qubit dataset, the unitaries are the $\hat{\mathrm{H}}$ and $\hat{\mathrm{K}}$ gates.

Just like the positive wavefunction, for the complex case we optimize the network parameters to minimize the contrastive divergence between the data and the RBM distribution. When measuring in multiple bases, the optimization now runs over the set of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$ and minimizes the sum of KL divergences between the data distribution $P(\mathbf{x}^{\boldsymbol{b}})$ and the RBM distribution $|\psi_{\boldsymbol{\theta}\boldsymbol{\phi}}(\mathbf{x}^{\boldsymbol{b}})|^2$ for each basis $\boldsymbol{b}$ appearing in the training dataset [101]. This is equivalent to minimizing the quantum relative entropy, an information measure between states [155, 156]

$$S(\phi|\sigma) = \operatorname{tr} \rho \log \phi - \operatorname{tr} \rho \log \sigma \tag{4.43}$$

which shares many properties of the KL divergence.

For example, if a given training sample is measured in the basis (x, z), we apply the appropriate unitary rotation $\boldsymbol{\mathcal{U}} = \hat{\mathrm{H}}_0 \otimes \hat{\mathcal{I}}_1$ to the RBM state before collecting the gradient signal.

Similar to the case of positive wavefunction, we generate the complete Hilbert space to compute the fidelity and KL divergence for a benchmark example. In Fig. 4.6 we show the total KL divergence and the fidelity with the true two-qubit state during training. After successfully training a model, we can once again compute expectation values of physical observables, as discussed in Section 4.3.

4.5 Summary

In this Section, we have shown the feasibility of using the RBM as a generative model for both positive and complex wavefunctions. From a set of projective measurements, we

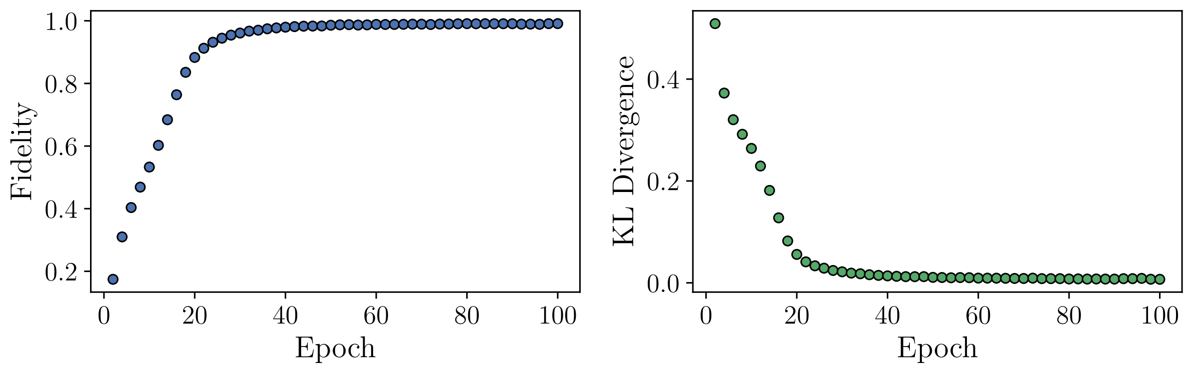


Figure 4.6: Training a complex RBM on random two-qubit data. We show the fidelity (left), and KL divergence (right), as a function of the training epochs.

train an RBM which can estimate observables including entanglement entropy. RBMs are trained with contrastive divergence, which is slow because it relies on MCMC. Further, contrastive divergence is not strictly equivalent to maximum likelihood. However, RBMs are essentially an Ising problem and so are very familiar to physicists and serve as a useful starting point for generative modelling.

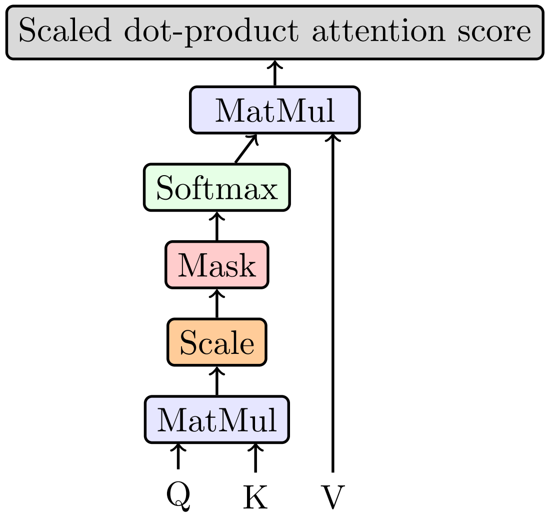
4.6 Transformers for state reconstruction

RBMs have the benefit of being physically motivated as an Ising model, however they have seen a rapid decline in use with machine learning practitioners. One of the most successful models in natural language processing is the transformer architecture. Introduced in 2017 by Vaswani et al. [157], the transformer uses a *self-attention* mechanism to achieve state-of-the-art results in many tasks. One of the main benefits of the transformer, as opposed to recurrent network models or RBMs, is that it lends itself to parallelization at a massive scale. For instance, Nvidia used the transformer architecture in Megatron-LM, an 8.3 billion parameter language model that was trained on 512 NVIDIA Tesla V100 GPUs [47]. Massive models have routinely bested language benchmarks such as the General Language Understanding Evaluation (GLUE) tasks [158, 159],

The transformer is built on a mechanism called self-attention. Self-attention take a vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ and computes a correlation s between all components, hence is a type of preprocessing to make the network more aware of the correlations between points. There are multiple ways to implement this attention mechanism. In the transformer, we focus on scaled dot-product self-attention. Instead of the usual $C_{ij} = \langle x_i x_j \rangle$ correlation matrix, we allow C_{ij} to be from the dot product of two different linear transformations of $\mathbf{x}$ called the query Q and key K. For each site $x_i \in \mathbf{x}$, there is a corresponding query and key vector. The mappings from $\mathbf{x}$ to Q, K is parameterized as a learning transformation

$$Q_j = \sum_i q_{ij}\ \mathrm{x}_j, \qquad K_i = \sum_i k_{ij}\ \mathrm{x}_i\,. \tag{4.44}$$

The dimension of the query and key vectors is not necessarily the same as $\mathbf{x}$, so we denote this as d. We can form a matrix by concatenating the Q_i vectors into a $d \times N$ matrix $\boldsymbol{Q} = [Q_1, Q_2, \ldots Q_N]$ and likewise for the keys $\boldsymbol{K} = [K_1, K_2, \ldots K_N]$. This lends itself well to parallel operations like matrix multiplication.



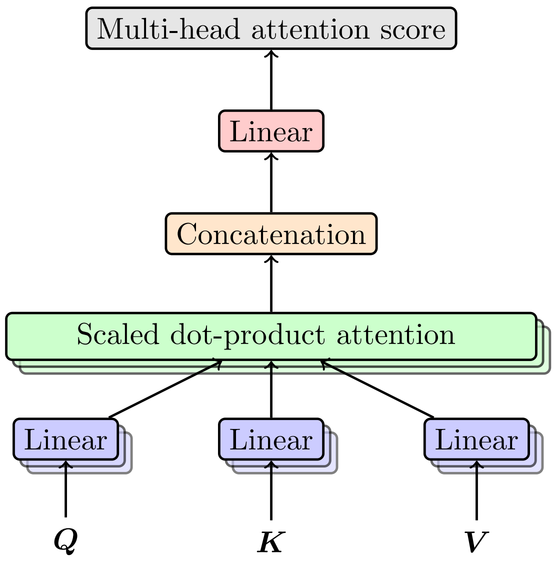


Figure 4.7: (a) A single attention mechanics. (b) Multi-head attention (c) the full transformer block.

Since Q and K contain the inputs x, taking a dot product between the matrices produces a type of correlation matrix. The scaled dot-product correlation matrix is

$$\boldsymbol{C} = \mathrm{softmax}\left(\frac{\boldsymbol{Q}^T\boldsymbol{K}}{\sqrt{d}}\right) \tag{4.45}$$

where the softmax function is

$$\text{softmax}\left(x_i\right) = \frac{e^{x_i}}{\sum_j e^{x_j}} \,. \tag{4.46}$$

The $\boldsymbol{Q}^T\boldsymbol{K}$ term acts like the correlation matrix by inducing pairwise all-to-all correlations. This is also similar to a n-body Jastrow factor using in quantum Variational Monte Carlo [12, 160]. In practice, the dot-product between the keys and values becomes larger as the dimension d increases.[2] This results in the saturation of the softmax function and hence extremely small gradients. The factor $1/\sqrt{d}$ attempts to mitigate this issue.

$$\text{Attention}(Q, K, V) = \boldsymbol{C}\boldsymbol{V} = \text{softmax}\left(\frac{\boldsymbol{Q}^T\boldsymbol{K}}{\sqrt{d}}\right) \tag{4.47}$$

to produce the score matrix, where we call $\boldsymbol{V} = [V_1, V_2, \ldots, V_N]$ the *values* defined with

$$V_j = \sum_i v_{ij}\ \mathrm{x}_j \,, \tag{4.48}$$

where in our case we take the dimension of the value vector V to be d, although generally it is an additional hyperparameter.

A key innovation from Vaswani et al. [157], is that we can use multiple attention mechanisms to essentially create an entire deep neural network with only attention. Multi-head attention is the concatenation of multiple attention scores

$$\text{Mulithead}(Q, K, V) = \text{concat}\left(\text{head}_1, \text{head}_2 \ldots\right) W \tag{4.49}$$

where $\text{head}_n = \text{Attention}(Q_n, K_n, V_n)$ and W is an additional parameterize linear transformation. A diagram of single, and multihead attention is shown in Fig. 4.7. Notice that a transformer block is built from the self-attention layer, and also involves batch normalization, dropout, a skip connection, and a fully connected layer.

For each spin $x_i \in \mathbf{x}$, the total score is given by value vector weighted by the attention matrix, both with learnable parameters. This gives a flexible architecture that can capture

[2]To see this, we can consider the example of vectors q and k composed of d elements that are independent and normally distribution with zero mean and variance σ^2. The dot product $q \cdot k$ also has zero mean, however the variance is a sum over each term q_ik_i so that the total variance is $d\sigma^2$. Hence, the standard deviation grows as $\sqrt{d}$.

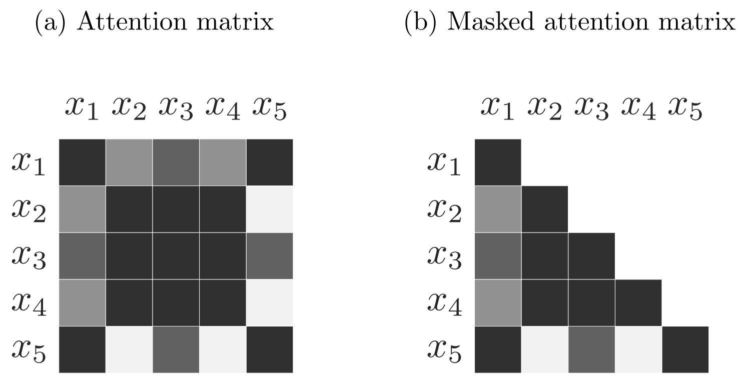

(a) Attention matrix
(b) Masked attention matrix
x_1 x_2 x_3 x_4 x_5
x_1 x_2 x_3 x_4 x_5

Figure 4.8: (a) Self-attention matrix for a five element vector $\mathbf{x}$. (b) Self-attention matrix with certain correlations masked out to allow the autoregressive property.

long-range correlations between any variables at the expense of additional matrix multiplications (Q, K, V). The transformer network faces a computational runtime of $O(dN^2)$ for a sequence of length N and an embedding dimension d. A more recent method has proposed approximating the scaled dot-product with a hash table to improve scaling to $O(dN \log(dN))$ [161]. In contrast, recurrent neural networks are of order $O(Nd^3)$.

The RBM drew samples from $p_{\boldsymbol{\theta}}$ using MCMC since the partition function Z_θ was intractable. In contrast, the transformer provides an explicit normalized probability for each sample. This is a very desirable quality for physicists and permits efficient autoregressive sampling.

In the transformer, the autoregressive property can be enforced by “masking out” any element in the upper triangle of the attention matrix

$$\text{Attention}_{\text{masked}}(Q, K, V) = \begin{cases} \text{Attention}(Q, K, V) & i \leq j \\ 0 & \text{otherwise} \end{cases} \tag{4.50}$$

as shown in Fig. 4.8a. This ensures that each variable x_i only depends on the preceding variables $x_{i-1}, \ldots, x_1$. This ancestral dependance is called the autoregressive property in Chapter 1, Section 1.1.5. This permits efficient autoregressive sampling which has the benefit of zero auto-correlation time. The caveat is that the network must be trained sufficiently. If the network converges to a poor optima, there may be implicit correlations present in the samples.

4.6.1 Fidelity for quantum states

Notice that in the σ^z basis, the wavefunction for the ground state TFIM, $\psi(\boldsymbol{x})$, contains only non-negative elements. Reconstructing the state can thus be seen as learning and sampling from $p(\boldsymbol{x})$. Because the TFI model provides exact probabilities for configurations, it makes it ideal to use as a benchmark since we can calculate the fidelity, $f(\psi, \psi_\theta) = |\langle\psi|\psi_\theta\rangle|^2$, between the neural network model and the exact solution. The fidelity provides an upper-bound on the error on all other observables $c|\langle O\rangle - \langle O_\theta\rangle| \leq 1 - f(\psi, \psi_\theta)$ for a constant c that depends on the specified observable. This provides a strict benchmark compared to other methods which only consider observables [162].

Computing the fidelity typically required exponential resources which makes it intractable for large systems. However, the fidelity can be approximated via importance sampling as

$$f = \sum_{\boldsymbol{x}} \sqrt{p(\mathbf{x})p_\theta(\mathbf{x})} \approx \frac{1}{M}\sum_{\tilde{\boldsymbol{x}}}^{M} \sqrt{\frac{p_\theta(\tilde{\mathbf{x}})}{p(\tilde{\mathbf{x}})}} \tag{4.51}$$

where $\tilde{\mathbf{x}}$ is importance sampled from $p(\mathbf{x})$. The exact probabilities $p(\mathbf{x})$ are computed using the method from Section. 1.1.4. The transformer produces $p_{\boldsymbol{\theta}}(\mathbf{x})$ for $\mathbf{x}$ in the training or test set. Alternatively, we can produce samples from the transformer and compute the exact amplitude. Both methods are equally viable for our purposes, although recall that the Monte Carlo requires evaluation of a determinant to get the amplitude of a configuration hence is $O(N^3)$, while the transformer is $O(N^2)$ but requires training data.

4.6.2 Results

We trained the transformer network on TFIM data for $N_s = 100,000$ measurements at the critical point $h = 1$. We used the Adam optimizer with an initial learning rate of 5×10^{-4} and a learning rate scheduler that decreases the learning rate by half every 10 epochs. Each network was trained 5 times, and the best one selected. We varied the number of attention heads and found that a single head was sufficient. Next, we varied the embedding dimension d, as shown in Fig. 4.9a. We also studied the reconstruction quality for various system sizes in Fig. 4.9b.

Further, we compared transformer to the RBM using the fidelity for $N = 16$ spins in Fig. 4.10. We find the transformer achieves better fidelity in the same number of epochs as the RBM. This could be due to different learning rates, or that the RBM minimized the contrastive divergences while the transformer directly minimizes the KL divergence.

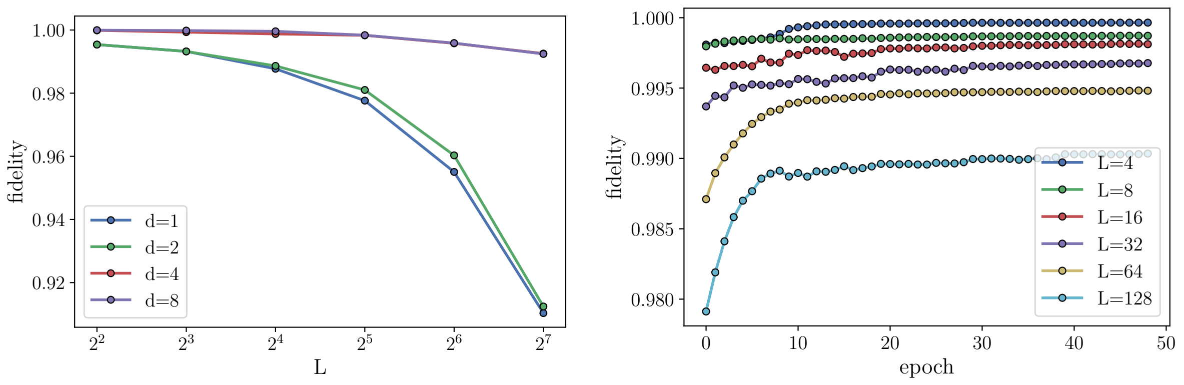


Figure 4.9: (a) The fidelity f, for $L = 32$ spins for various embedding dimension sizes d. (b) Fidelity improvement per epoch during training for various system sizes L.

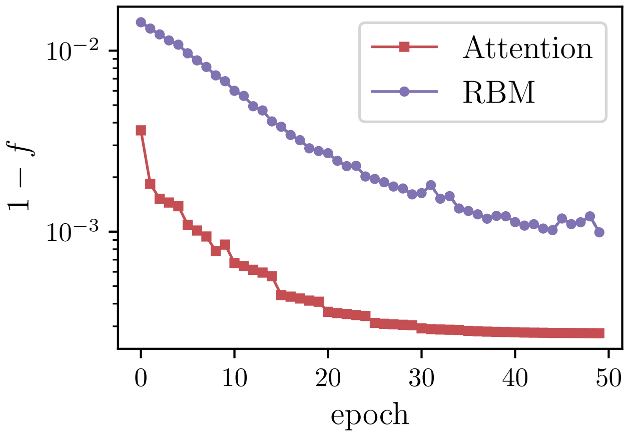


Figure 4.10: Monitoring the infidelity $1 - f$, for the RBM and self-attention based transformer for the ground state of the TFI model with $L = 16$ spins.

4.7 Conclusion

In this Chapter, we have explored quantum state reconstruction with neural networks. With a data set representing projective measurements of a quantum wavefunction, either produced by experiment or numerical methods, we can extract features and approximate the quantum state.

We mostly focused on the one-dimensional TFIM, a prototypical example of an interacting quantum many-body system. Using restricted Boltzmann machine (RBM), a classical Ising model themselves, or more recent self-attention models, we have shown the ability of these models to represent the ground state of the TFIM.

Once properly trained, RBMs or transformers can produce a new set of measurements, sampled from the model. These samples, generated in the reference basis, can be used to verify the training of the model against the original data set. More importantly, they can be used to calculate expectation values of many physical observables. In fact, any expectation value typically estimated by conventional Monte Carlo methods can be implemented as an estimator. Such estimators may be inaccessible in the reference basis, for example. Or, they may be difficult or impossible to implement in the setup for which the original data was obtained. This is particularly relevant for experiments, where it is easy to imagine many possible observables that are inaccessible, due to fundamental or technical challenges.

The techniques described in this paper can also be extended to reconstruct mixed states, via the purification technique described in Torlai et al. [102]. In addition, future techniques may include hybridization between machine learning and other well-established methods in computational quantum many-body physics, such as variational Monte Carlo and tensor networks [130].

5

Quantum-inspired variational methods

In this Chapter, we introduce a variational wavefunction for many-body ground states that involves imaginary time evolution with two different Hamiltonians in an alternating fashion with variable time intervals. The ansatz is inspired by the quantum approximate optimization algorithm [163, 164]. We successfully apply the ansatz on the one- and two-dimensional transverse-field Ising model and systematically study its scaling for the one-dimensional model at criticality. The total imaginary time required scales logarithmically with system size, in contrast to the linear scaling in conventional Quantum Monte Carlo. We suggest this is due to unique dynamics permitted by alternating imaginary time evolution, including exponential growth of bipartite entanglement. For generic models, the superior scaling of our ansatz potentially mitigates the sign problem at the expense of having to optimize variational parameters.

5.1 Introduction

Imaginary time plays a prominent role in multiple branches of physics, including cosmology, statistical mechanics and quantum field theory. The seemingly simple replacement of real time, t, with its imaginary counterpart, $\tau = -it$, leads to fundamental connections between quantum theory and statistical mechanics [165]. Such connections enable the efficient simulation of many quantum systems using quantum Monte Carlo techniques [166, 167, 168]. However, for many physically interesting models, these methods suffer from the prohibitive 'negative sign problem' [169, 153], which requires an exponential amount of computational resources to obtain reasonable accuracy for quantum many-body systems. Many outstanding problems in condensed matter, such as those involving high temperature superconductors or topologically ordered phases, require an understanding of complex interacting models which are unsolved with present techniques.

One class of Monte Carlo methods that can avoid the sign problem are so-called variational Monte Carlo (VMC) methods [170, 171, 32, 172, 173, 12]. In VMC, one assumes a sufficiently general trial state that depends on adjustable parameters. These parameters are then chosen to minimize the energy with respect to the given Hamiltonian. Finding an effective trial state such as Jastrow [12], matrix product states [174, 175, 176], or neural network states [13, 177, 100, 178], can result in efficient simulation of interacting quantum systems. The key to the success of these techniques is a well-chosen ansatz that reflects the properties of the target phase and the existence of a viable optimization scheme [179, 180, 181, 182].

The recent advent of quantum computers and simulators has motivated the development of new variational approaches [183]. Such variational quantum algorithms involve applying a sequence of unitary operators, parameterized by several variables onto an easy-to-prepare initial state. The variables are chosen to optimize a given cost function involving the resulting wavefunction. For example, in the quantum approximate optimization algorithm (QAOA) [163, 164, 184, 185, 186, 187, 188] the cost function is a classical Hamiltonian encoding a combinatorial optimization problem, and the variational wavefunction is prepared by alternating between evolving with the Hamiltonian and a transverse field. The evolution times constitute variational parameters that are optimized to minimize the Hamiltonian cost function. This variational approach has been generalized for preparing both strongly correlated and highly-entangled states on near-term quantum devices [189, 190, 191].

Motivated by the success of such variational approaches, in this Chapter, we propose a variational ansatz for ground states of quantum many-body systems which involves sequentially evolving with different Hamiltonians in imaginary time. In contrast to real time evolution with local Hamiltonians, which is limited by Lieb-Robinson bounds on the growth of correlation functions, imaginary time evolution does not have this constraint and can exhibit remarkable efficiency in traversing Hilbert space.

As proof of concept, we demonstrate the efficiency of our ansatz in representing the ground state of the transverse field Ising model at criticality. Whereas standard projector methods require imaginary time scaling with system size L to reach the critical ground state, we show numerically that our ansatz requires time scaling logarithmically with L. Furthermore, we analyze how entanglement grows after each imaginary time operation in our ansatz, and we find an exponential growth that is a unique feature of imaginary time dynamics. We conclude by demonstrating that the ansatz continues to perform well in the presence of integrability-breaking perturbations, and we mention generalizations of our ansatz to other models, including those with sign problems. We envision the main purpose of this ansatz to be an efficient trial wavefunction for quantum many-body physics on (classical) computers, however, it is possible that one can also implement such imaginary time evolution natively on a quantum computer [192, 193].

TO begin, we summarize the quantum approximate optimization algorithm (QAOA) in Section 5.2. Next, in Section 5.3, we study the imaginary time version of QAOA with an exact free-fermion mapping. In Section 5.4, we implement a hybrid variational/path integral Monte Carlo that corresponds to the imaginary time ansatz.

5.2 Quantum approximate optimization algorithm

A promising application of quantum computers is combinatorial optimization problems. Optimization problem will typically involve an objective over bit strings of length N. The solution can be encoded in the ground state of a cost Hamiltonian H_C, which is diagonal in the computational basis.

The quantum adiabatic algorithm proposes a method to find the ground state of the cost Hamiltonian H_C by preparing a known state of a simple Hamiltonian, H_X and evolving adiabatically from H_X to H_C [194, 195, 196]. Typically, H_X is a transverse magnetic field on each qubit so the initial state is the uniform superposition over all states $|+\rangle$. The simplest adiabatic path is $H(t) = (1 - \frac{t}{T})H_X + \frac{t}{T}H_C$, where t is time and T the total time. At $t = 0$, the state is $|+\rangle$, and at $t = T$, the state is the target state $|\psi\rangle$. For finite T, there is a chance that the system will jump to an excited state. To keep this probability less than ϵ, the evolution time must be $T \geq \frac{1}{\Delta_{\min}^2}$, where $\Delta_{\min} = E_1 - E_0$ is the minimum energy gap between the ground and first excited states along the path. Unfortunately, in certain problems the gap can be exponentially small [197]. This means the adiabatic algorithm could be even worse than a classical brute force solution in certain cases.

Hogg and Portnov (and subsequently Farhi et el.) realized that to implement the time-evolution required by the adiabatic algorithm on a gate-based quantum computer, one needs to break down each time step use the Suzuki-Trotter formula [163, 164]. The Suzuki-Trotter formula breakdown each small-time evolution with $H(t)$ into the product of individual evolutions

$$e^{-iH(t)dt} \approx e^{-i(1-t)H_X dt} e^{-iH_C dt} \tag{5.1}$$

so that the entire path for P steps takes the form

$$|\Psi_{\boldsymbol{\alpha},\boldsymbol{\beta}}\rangle = e^{-i\int H(t)\,dt}|+\rangle \approx e^{-i\beta_P H_X}e^{-i\alpha_P H_C}\cdots e^{-i\beta_1 H_X}e^{-i\alpha_1 H_C}|+\rangle \tag{5.2}$$

where α_p, β_p are variational parameters what define the trotterized path. Finding the optimal parameters by minimizing $\langle\Psi_{\boldsymbol{\alpha},\boldsymbol{\beta}}|H_C|\Psi_{\boldsymbol{\alpha},\boldsymbol{\beta}}\rangle$ gives a discrete version of the adiabatic algorithm known as QAOA. A schematic diagram is shown in Fig. 5.1.

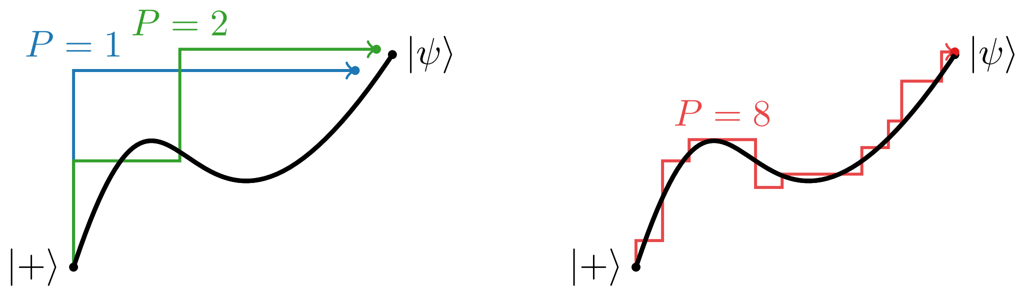


Figure 5.1: QAOA for $P = 1, 2, 8$. Note that the $P = 1$ solution might not be able to exactly reach the target state, but it can be close. Each P systematically improves performance (if optimization is possible).

QAOA briefly achieved state-of-the-art results on the MaxE3LIN2 problem [184] until a classical version was found [198]. The algorithm has also been shown to re-derive important algorithms such as Grover's search [199]. However, optimization of many variational parameters can be difficult since there is no direct way to compute gradients. With real quantum hardware, one is restricted to using the parameter-shift rule which means running similar circuits multiple times [200]. In simulators, one can use reverse-mode automatic differentiation to remain efficient [62]. However, a general solution to the quantum gradient problem is unknown.

Furthermore, QAOA may not be robust to noise for depth $P > 3$ [201, 202] on current devices. It was also suggested that QAOA is not optimal for MaxCut on triangle-free graphs [203, 204, 205]. Finding the border between quantum advantage and classical computation is very important to the future of quantum computing.

In the remainder of this Chapter, we introduce an imaginary-time version of QAOA that is efficient on classical computers and allows the calculations of gradients. We show better scaling compared to quantum QAOA and explain its success in terms of entanglement growth. We implement both an exact method and a stochastic Monte Carlo approach.

5.3 Variational imaginary time ansatz

Many Hamiltonians are naturally a linear combination of two components $H = H_A + gH_B$, where $H_{A,B}$ are individually tractable to analyze. Examples include transverse field Ising, Hubbard, and the J_1–J_2 model. Motivated by the QAOA procedure, we consider the following variational imaginary time ansatz (VITA) for the ground state of H:

$$|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle = \mathcal{N} e^{-\beta_P H_B} e^{-\alpha_P H_A} \cdots e^{-\beta_1 H_B} e^{-\alpha_1 H_A} |\psi_0\rangle \tag{5.3}$$

where P is the number of pairs of variational parameters $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_P), \boldsymbol{\beta} = (\beta_1, ...\beta_P)$, $|\psi_0\rangle$ is an initial state, and $\mathcal{N}$ is a normalization factor. We further define the total imaginary time $\tau = \frac{1}{2}\sum_{p=1}^{P}(\alpha_p + \beta_p)$. A circuit representation is shown in Fig. 5.2.

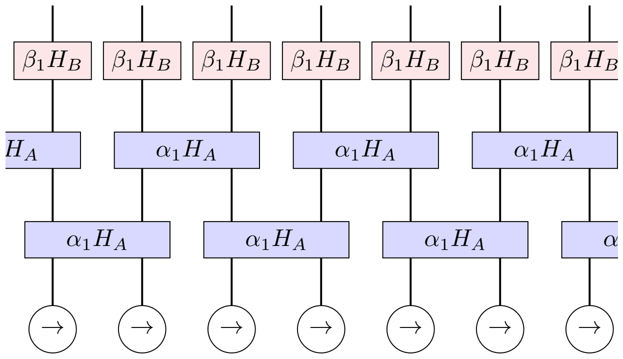


Figure 5.2: Circuit representation of the trial state $|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle$ for $P = 1$. Each box denotes imaginary time evolution with the enclosed Hamiltonian.

While VITA is applicable to any Hamiltonian, in specific cases there is explicit physical motivation for considering such an ansatz. For example, for the fermionic Hubbard model, the $P = 1$ ansatz reduces to Otsuka's generalization of the Gutzwiller variational wavefunction [172, 170, 171, 173], which seeks to balance single occupancy per site with itinerancy. The $P \leq 3$ case has been considered in Ref. [206, 207, 208] for the two-dimensional Hubbard model, but a systematic analysis of how its performance scales with system size and P was not carried out. A related variational approach for the Hubbard model has also been considered in Ref. [209].

The standard projector method for attaining the ground state of H involves evaluating $e^{-\tau H}|\psi_0\rangle$ for $\tau \geq 1/\Delta$ where Δ is the spectral gap. This can be decomposed via trotterization into a sequence of the VITA form, with parameters $\alpha_p = \beta_p = \tau/2P$ for large P. This guarantees that VITA can exactly represent the ground state in the $P \to \infty$ limit. However, the projector method is especially expensive for critical systems where $\Delta \sim 1/L$, and hence τ scales polynomially with L. One can consider Eq. (5.3) as a non-uniform trotterization with large (and variable) times steps. We will show that remarkably high fidelity can be attained even with τ that is *exponentially smaller* compared to the aforementioned estimate from the standard projector method.

The advantage from the bang-bang procedure of QAOA is that the total time needed to evolve to the ground state may be shortened substantially. In the case of real QAOA, the addition of the bang-bang type procedure allows for great improvement over the adiabatic method. In particular, circuits can target states with high accuracy despite only possessing few operators [190, 191]. These low-depth circuits are of particular interest because they are feasible on near-term quantum devices.

We treat the problem from finding the ground state as a variation Monte Carlo (VMC) problem. To discover the optimal parameters, $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for a fixed P, we take the approach of minimizing the energy,

$$E_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \frac{\langle \psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})|H|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle}{\langle \psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle} \tag{5.4}$$

rather than the variance. Note that Eq. (5.4) contrasts the usual use of QAOA in combinatorial optimization problem where one encodes the solution into the ground state of the problem Hamiltonian H_C.

We first present some general considerations of why such an ansatz may be efficient. It is useful to first compare with the real-time analogue, which are QAOA-type circuits involving alternating real-time evolution between two Hamiltonians. The Lieb-Robinson bound dictates that real-time evolution with local Hamiltonians can generate correlations only within a light cone, and thus there are lower bounds on the time it takes to prepare highly correlated states starting from unentangled product states. For example, in one dimension, the total time to prepare the GHZ ("cat") state $\frac{1}{\sqrt{2}}(|11\ldots1\rangle + |00\ldots0\rangle)$ scales at least linearly with system size L [210, 190]. In contrast, by evolving with the GHZ parent Hamiltonian $H_{ZZ} = -\sum_{i=1}^{L} Z_i Z_{i+1}$ in imaginary time (Z is the Pauli-Z matrix), the GHZ state can be prepared with imaginary time scaling as $\log L$.

We test VITA on the transverse-field Ising model (TFIM)

$$H = H_{ZZ} + hH_X \equiv -\sum_{i=1}^{N} Z_i Z_{i+1} - h\sum_{i=1}^{N} X_i \tag{5.5}$$

with periodic boundary conditions on a system with N spins We use Z, X to denote the Pauli matrices and h for the transverse field strength. Our ansatz in this case starts from the paramagnetic ground state of H_X, $|+\rangle$ and alternates between $H_A = H_{ZZ}$ and $H_B = H_X$.

The Ising chain can be mapped to free fermions via the Jordan-Wigner transformation [211] which allows for the efficient evaluation of Eq. (5.3). We can thus optimize our ansatz for very large system sizes and several pulses. This allows us to properly characterize how efficient the VITA ansatz is without introducing sampling error.

5.3.1 Jordan-Wigner transformation

The TFIM conveniently admits a solution as free fermions by using a Jordan-Wigner transformation [211, 212]. The problem of L Ising spins can be mapped to $L/2$ independent spin-$\frac{1}{2}$ fermions by introducing the fermionic operators

$$a_j = e^{i\pi\phi_j} S_j^- \qquad a_j^\dagger = e^{-i\pi\phi_j} S_j^+ \qquad \phi_j = \sum_{i=1}^{j-1} S_i^+ S_i^- \tag{5.6}$$

where $S_j^\pm = (Y_j \pm i Z_j)/2$ are the raising/lowering operators and ϕ_j introduces a non-locality to the transformation. Note that this differs slightly from the usual convention where the TFIM in defined with the Z and X matrices interchanged.

We confirm that these operators are indeed fermionic by checking the anti-commutation relations,

$$\left\{a_i^\dagger, a_j^\dagger\right\} = \left\{a_i, a_j\right\} = 0 \tag{5.7}$$

$$\left\{a_i, a_j^\dagger\right\} = \delta_{i,j}\,. \tag{5.8}$$

To recover the original spins from the fermionic representation, one can use the inverse transformation

$$S_j^+ = a_j^\dagger e^{-i\phi_j} \qquad S_j^- = a_j e^{i\phi_j} \qquad \sigma_j^x = 2a_j^\dagger a_j - 1\,. \tag{5.9}$$

The Hamiltonian operators of Eq. (5.5) are then

$$H_X = \sum_{j=1}^{N} \left(2a_j^\dagger a_j - 1\right) \tag{5.10}$$

$$H_{ZZ} = \sum_{j=1}^{N-1} a_j^\dagger a_{j+1} + a_j a_{j+1} - \left(a_N^\dagger a_1 + a_N a_1\right) G + \text{h.c.} \tag{5.11}$$

$$G = \exp\left(i\pi \sum_{j=1}^{N} a_j^\dagger a_j\right) = \exp\left(i\pi N_f\right) \tag{5.12}$$

with N_f the total number of fermions, and h.c. denotes the Hermitian conjugate. The gauge operator $G = 1$ in the case of an even number of fermions, while $G = -1$ for an odd number of fermions. We will only consider the case of even N, with periodic boundary conditions.

Next, we Fourier transform the operators with

$$c_k^\dagger = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} e^{i\theta_k n} a_n^\dagger, \qquad \theta_k = \frac{(2k+1)\pi}{L}, \qquad k = 0, \cdots, L-1\,. \tag{5.13}$$

Setting $\theta_k = \frac{(2k+1)\pi}{L}$, the Hamiltonian Eq. (5.5) in momentum space becomes

$$H_X = \sum_{k=0}^{L-1} \left(2c_k^\dagger c_k - 1\right) \tag{5.14}$$

$$H_{ZZ} = 2 \sum_{k=0}^{L-1/2} \cos\theta_k \left(c_k^\dagger c_k + c_{-k}^\dagger c_{-k}\right) + i \sin\theta_k \left(c_k^\dagger c_{-k} + c_k^\dagger c_{-k}\right)\,. \tag{5.15}$$

Using the inverse transformation from Eq. (5.9), we can write

$$H_{X,k} = 2\tilde{X} \tag{5.16}$$

$$H_{ZZ,k} = 2\left(\cos\theta_k \tilde{Y} + \sin\theta_k \tilde{Z}\right)\,, \tag{5.17}$$

where we use $\tilde{Z}$ to denote that it is not in the original position space of the problem, but rather in momentum space acting on a ket $|\chi_k\rangle = |0_k 0_{-k}\rangle + |1_k 1_{-k}\rangle$.

The VITA ansatz in Eq. (5.3) then amounts to

$$|\psi_P(\boldsymbol{\alpha}, \boldsymbol{\beta})\rangle = \bigotimes_{k=1}^{N/2} \prod_{i=1}^{p} e^{-\alpha_i H_{X,k}} e^{-\beta_i H_{ZZ,k}} |\chi_k\rangle \ , \tag{5.18}$$

acting on the single-site states $|\chi_k\rangle$.

5.3.2 Exact solution

The TFIM is an exactly solvable model via the Jordan-Wigner transformation. To diagonalize the combined Hamiltonian, we use a Bogoliubov transformation

$$\begin{pmatrix} c_k \\ c_{-k}^\dagger \end{pmatrix} = \begin{pmatrix} u_k & v_k \\ -v_k^* & u_k \end{pmatrix} \begin{pmatrix} \eta_l \\ \eta_{-k}^\dagger \end{pmatrix}, \tag{5.19}$$

so that

$$H = h - \sum_{k>0} E_k \left(\eta_k^\dagger \eta_k + \eta_{-k}^\dagger \eta_{-k}^\dagger - 1 \right) + \zeta_0 c_0^\dagger c_0 + \zeta_\pi c_\pi^\dagger c_\pi \,. \tag{5.20}$$

The u_k, v_k which satisfy this are

$$u_k = \sqrt{\frac{E_k + \zeta_k}{2E_k}}, \qquad v_k = i\sqrt{\frac{E_k - \zeta_k}{2E_k}}, \tag{5.21}$$

$$E_k = \sqrt{J^2 + h^2 + 2Jh\cos k}, \tag{5.22}$$

$$\zeta_k = -h - J\cos k \,. \tag{5.23}$$

Both $H_Z Z$ and H_X preserve parity, so if considering even N, the ground state corresponds to zero or two fermions. The ground state of the total Hamiltonian is therefore the Bardeen–Cooper–Schrieffer (BCS) state

$$|\text{BCS}\rangle = \prod_{k=1}^{N/2} \left(u_k + v_k c_k^\dagger c_{-k}^\dagger \right) |0\rangle \ . \tag{5.24}$$

Having the exact state and energy allows us to benchmark VITA by comparing the energy and fidelity of the target and variational states.

5.3.3 Optimization

The optimization landscape for VITA$_1$ and $L = 10$ spins is shown in Fig. 5.3. We use differential evolution (adaptive/rand/1/bin) to find the minimum of the energy from Eq. (5.4) [213]. In our experience, VITA$_1$ has a convex landscape making optimization trivial. For higher P, it is difficult to verify the convexity, but we suspect it is not convex and optimization becomes difficult.

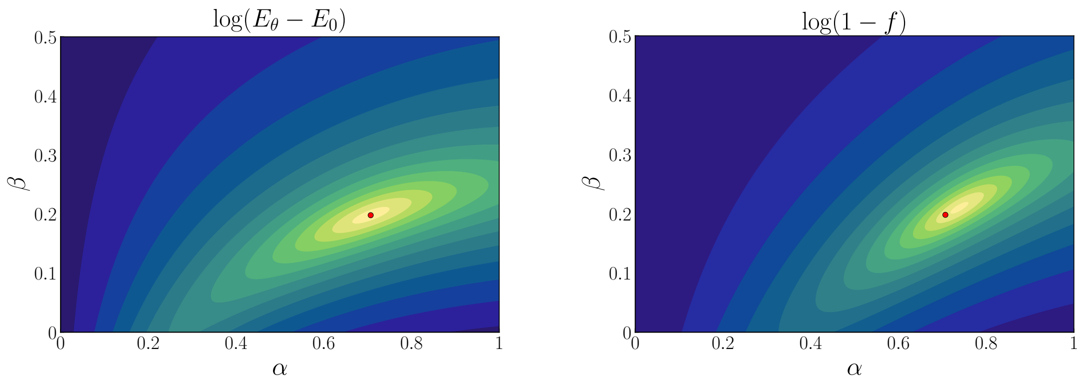


Figure 5.3: (a) Energy landscape for $P = 1$ and $L = 10$ spins. (b) Fidelity landscape. For visualization purposes, the colour represents the logarithm of the energy difference between the exact energy and the variational energy.

We first focus on approximating the critical ground state of H at $h = 1$. Figure 5.4a shows the relative error in energy $\epsilon_{\text{rel}} = |(E_P(\boldsymbol{\alpha}, \boldsymbol{\beta}) - E_{\text{exact}})/E_{\text{exact}}|$ where E_{exact} is the exact ground state energy at the critical point, for various P. Evidently, increasing P dramatically improves the accuracy in the energy, even for large system sizes.

Since the exact ground state for the TFIM is known, we also compare the fidelity, f, of the optimized trial state with the target state. The error in fidelity, $1 - f \equiv 1 - |\langle \psi_{\text{exact}} | \psi_P(\boldsymbol{\alpha}, \boldsymbol{\beta}) \rangle|^2$, is shown in Fig. 5.4b for various P, L. The efficiency is quite remarkable; for example, for $L = 64$, $P = 2$ is already sufficient to approximate the critical state to within around 10^{-4} in relative energy and 10^{-2} in fidelity. Recall that the error in fidelity provides an upper-bound for the error in any observable[1]. In Fig. 5.5, we show the

[1]If a state $|\psi\rangle$ is within ϵ fidelity with $|\phi\rangle$ then the difference in expectation values of an observable O is bounded by

$$\left| \langle O \rangle_\psi - \langle O \rangle_\phi \right| \leq 2c \left(\sqrt{\epsilon(1-\epsilon)} + \epsilon \right). \tag{5.25}$$

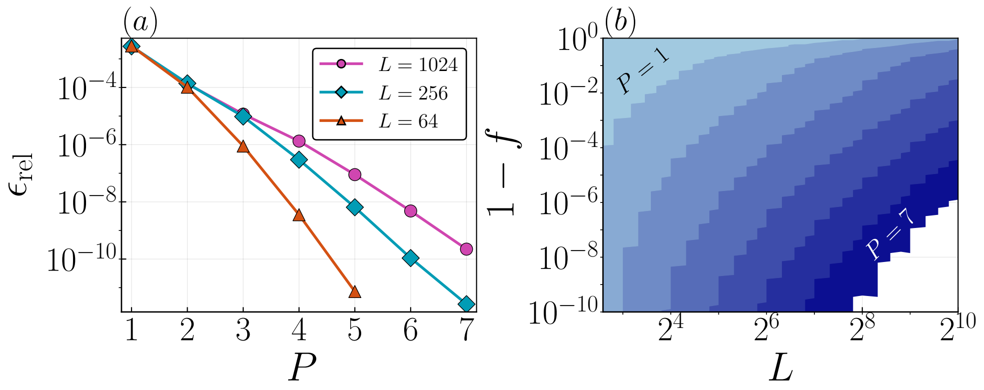


Figure 5.4: (a) Relative error in energy, ϵ_{rel} between the exact ground state energy, E_{exact}, and the energy of the optimized trial wavefunction $E_P(\boldsymbol{\alpha}, \boldsymbol{\beta})$. (b) Number of pulses P needed to obtain a desired accuracy in the fidelity, f, for a given system size, L. The white region in (b) was not computed in the present study.

energy, fidelity and time τ for each P, for various system sizes L.

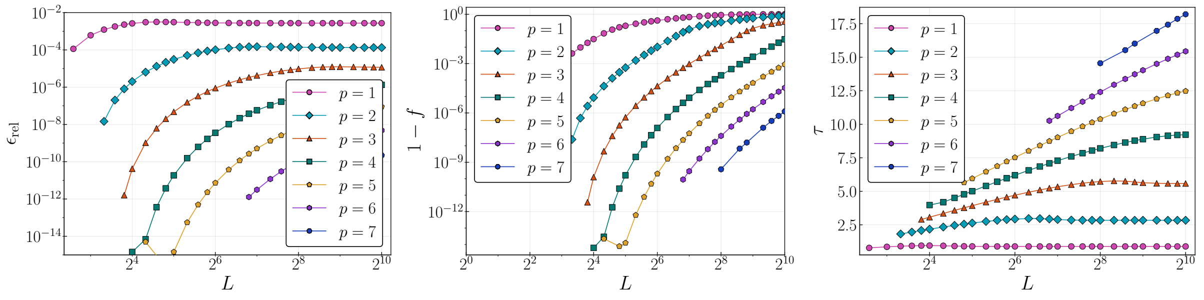


Figure 5.5: (a) Relative error in energy, $\epsilon_{\text{rel}} = |(E_P(\boldsymbol{\alpha}, \boldsymbol{\beta}) - E_{\text{exact}})/E_{\text{exact}}|$ between the VITA$_p$ ansatz for various system sizes L. (b) $1-f$, the infidelity of the ground state and the optimized state $|\psi_P(\boldsymbol{\alpha}, \boldsymbol{\beta})\rangle$. (c) Total imaginary time $\tau = \frac{1}{2}\sum_{p=1}^{P}(\alpha_p + \beta_p)$ for the optimized ansatz.

5.3.4 Scaling

To compare directly with the projector method, we also investigate the total imaginary time $\tau \equiv \frac{1}{2}\sum_{p=1}^{P}(\alpha_p + \beta_p)$ required to achieve a target fidelity. Motivated by the $P \propto \log L$ scaling for achieving a target fidelity, we propose a scaling form of $1 - f = G(\tau(\log L)^{-\nu})$

for some exponent ν. In Fig. 5.6, we perform a scaling collapse for $L \in [4, 6, \ldots, 1024]$, and $P \in [1, \ldots, 7]$ and find the optimal exponent $\nu = 2.3 \pm 0.1$. This logarithmic scaling is an exponential speedup compared to the linear scaling of the projector method, $1 - f = F(\tau L^{-1})$ [214].

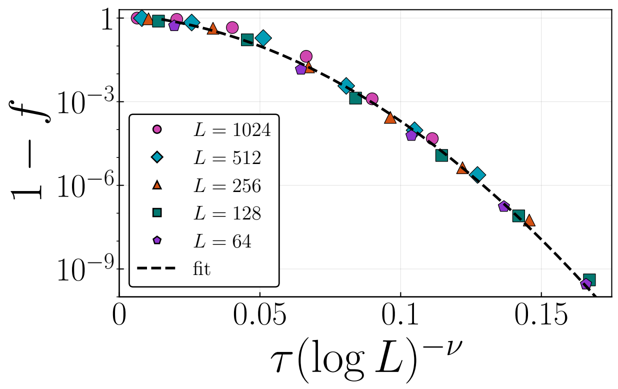


Figure 5.6: Collapse of the infidelity $\log(1 - f) = G(\frac{\tau}{(\log L)^{\nu}})$ with $\nu = 2.3$. The fit is a power law $\log(1 - f) = 175\, x^{1.85}$ with $x = \frac{\tau}{(\log L)^{\nu}}$.

We typically find optimizing $P = 1$ converges rapidly, while higher-P becomes more difficult. Optimal parameters for a fixed P vary smoothly with system size and field strength, which makes inferring 'nearly' optimal parameters easy (see Fig. 5.7). This result is highly similar to the patterns in optimal parameters found in Ref. [185] for QAOA. Notice that for small $h \sim 0$ the total time τ diverges linearly like the projector method. The total value for each parameter, with the exception α_1 as peaks just below the critical point $h = 1$. This is reminiscent of the scaling of parameters in an RBM for quantum state tomography [162].

5.3.5 Entanglement entropy

Because the TFIM is dual to a free system, all correlations, and hence the entanglement entropy (EE), can be determined from the two-point functions by Wick's theorem [215, 216, 217]. It is convenient to introduce Marjorana fermions $a_{2n} = i(c_n - c_n^\dagger)$, $a_{2n-1} = c_n + c_n^\dagger$

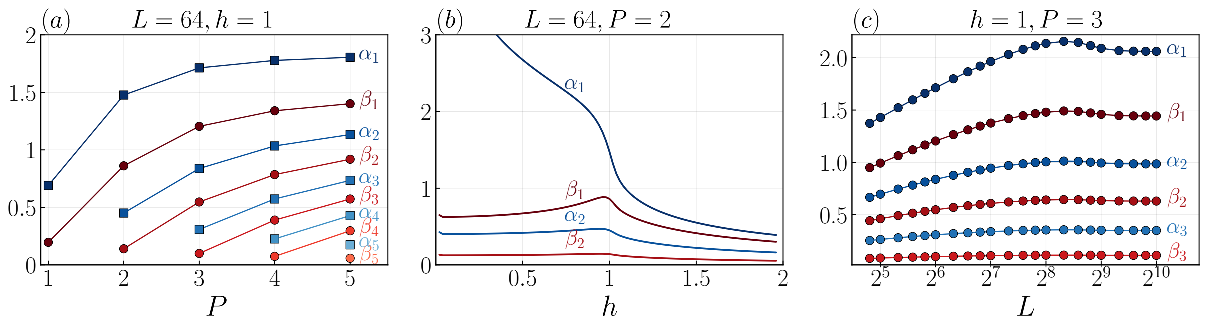


Figure 5.7: Optimal parameters $(\boldsymbol{\alpha},\boldsymbol{\beta})$ found using the Jordan-Wigner method for $L=64$ spins for (a) the critical point $h=1$ with $P=1,...,5$, (b) Optimal parameters for $P=2$ for various h, (c) scaling of optimal parameters with L at $h=1$. All plots are generic for any P and system size L.

with the $2L\times 2L$ correlation matrix $\langle a_n a_m\rangle = M_{nm} = \delta_{nm} + i\Gamma_{nm}$ given by

$$\Gamma_{ij} = \begin{pmatrix} \Pi_0 & \Pi_1 & \cdots & \Pi_{L-1} \\ \Pi_{-1} & \Pi_0 & & \vdots \\ \vdots & & \ddots & \vdots \\ \Pi_{1-L} & \cdots & \cdots & \Pi_0 \end{pmatrix}, \qquad \Pi_n = \begin{pmatrix} 0 & g_n \\ -g_{-n} & 0 \end{pmatrix} \tag{5.26}$$

with

$$g_n = \langle a_n a_0\rangle = \langle c_n^\dagger c_0^\dagger\rangle + \langle c_n^\dagger c_0\rangle - \langle c_n c_0\rangle - \langle c_n c_0^\dagger\rangle\ . \tag{5.27}$$

The correlation functions can be found from their Fourier transforms,

$$\langle c_k c_{-k}\rangle = u_k v_k^* \tag{5.28}$$
$$\langle c_k^\dagger c_{-k}^\dagger\rangle = u_k v_k^* \tag{5.29}$$
$$\langle c_k^\dagger c_k\rangle = v_k^2 \tag{5.30}$$
$$\langle c_k c_k^\dagger\rangle = u_k^2 \tag{5.31}$$

Using these expressions to compute g_n, we have

$$g_n = \frac{2}{L}\sum_{k=0}^{(L-1)/2} \left(2\tilde{u}_k\tilde{v}_k \sin n\theta_k + \left(\tilde{u}_k^2 - \tilde{v}_k^2\right)\cos n\theta_k\right) \tag{5.32}$$

where $\tilde{u}_k, \tilde{v}_k$ are real numbers that characterize the state. For the exact ground state, the above formula simplifies to

$$g_n = \frac{2}{L} \sum_{k=0}^{(L-1)/2} \frac{h \cos n\theta_k + J \cos(n+1)\theta_k}{E_k} \,. \tag{5.33}$$

The entanglement entropy of a region A can be obtained by restricting the correlation matrix Γ to a region A of size L_A. The eigenvalues of the reduced Γ_A, denoted with ν_m, contribute equally to the entanglement entropy through

$$S(L_A) = -\sum_{\nu} \left(\frac{1+\nu_m}{2}\right) \log_2 \left(\frac{1+\nu_m}{2}\right) + \sum_{\nu} \left(\frac{1-\nu_m}{2}\right) \log_2 \left(\frac{1-\nu_m}{2}\right) \tag{5.34}$$

This method scales polynomially in the system size L, allowing the study of systems with hundreds of spins. Figure 5.8 shows that the EE of our ansatz converges to the Cardy formula [218] as P increases.

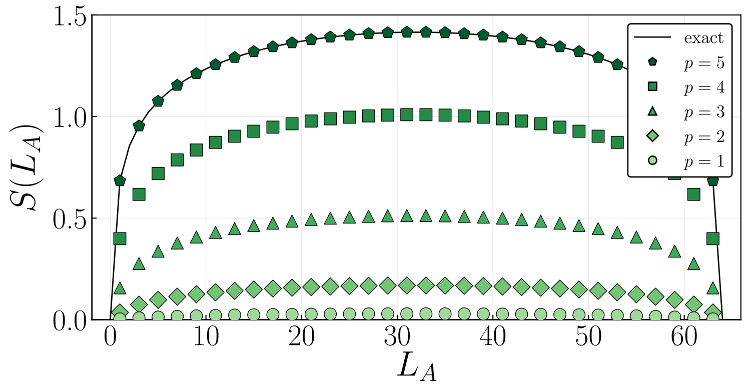


Figure 5.8: Entanglement entropy as a function of subsystem size L_A for intermediate p states in a depth $P = 5$ ansatz for $L = 64$ spins.

While there is no Lieb-Robinson bound limiting the rate for generating long-range correlations in our ansatz, entanglement considerations provide lower bounds on the iterations P required to prepare the critical state. Ignoring normalization, imaginary time evolution with a local Hamiltonian can be represented by a (non-unitary) quantum circuit; each iteration of our ansatz corresponds to three layers shown in Fig. 5.2. After P pulses, the bipartite entanglement entropy between the left and right halves the system, hereafter abbreviated EE, can be attained by bisecting the circuit through P bonds. Hence, EE after

P iterations is at most $P \log D$, where D is the Schmidt rank (number of singular values) upon decomposing a single two-qubit imaginary time operator.

In order to generate the EE of the critical state, which scales as $S \propto \log L$, we need the number of pulses scaling at least as $P \propto \log L$. We observe that Fig. 5.4b is consistent with this scaling form. For example, for a target fidelity error of 10^{-10}, each additional P in the ansatz can represent a system approximately twice as large.

The entanglement dynamics in imaginary time evolution can be considerably different from its real time counterpart. For real time evolution, EE across a bipartition can increase only by acting with an operator supported on both sides of the partition. If the circuit in Fig. 5.2 were unitary, any increase in EE from one layer to the next would be bounded by a constant depending on the two-qubit unitary but not on the state being acted on (see the "small incremental entangling theorem" [219]). In contrast, even imaginary time operators acting on one side of the bipartition can generate entanglement across the cut. As a very simple example involving two spins, the action of $e^{-\beta Z_1}$ on $\eta_+|11\rangle + \eta_-|00\rangle$ can increase EE as long as $|\eta_+| > |\eta_-| > 0$. This illustrates that the more entangled the initial state, the more imaginary time evolution can change the entanglement. The change is not simply bounded by a state-independent constant. This allows in principle an exponential growth of EE, as long as the total EE after P steps lies below $P \log D$.

Our ansatz exhibits such dynamics. We take the $P = 5$ ansatz and analyze the EE of the states at intermediate steps, p, of our protocol using the technique of [215, 216, 217]. We find that the EE increases exponentially with imaginary time (Fig. 5.9a). Moreover, for every intermediate state, we plot the mutual information ($S_A + S_B - S_{AB}$) between two spins A, B as a function of their separation (Fig. 5.9b). The power law decay for every step is in stark contrast to any local real time evolution and illustrates the ability of imaginary time evolution to generate long-range correlations. We find that under imaginary-time evolution, entanglement starts to grow immediately following a local quench, in contrast to real-time evolution [218, 220] where it takes a time proportional to ℓ before growing, ℓ being the distance between the location of the local quench and the entanglement cut.

5.3.6 Summary

In this Section, we showed the ideal scaling of the VITA ansatz from the TFIM by using the mapping to free-fermions. We found that remarkably even $P = 1$, can get within 2% of the exact energy regardless of system size. Moreover, the scaling of the total time τ was favourable as it scaled with $(\log L)^{2.3}$. The parameters also exhibited smooth dependence

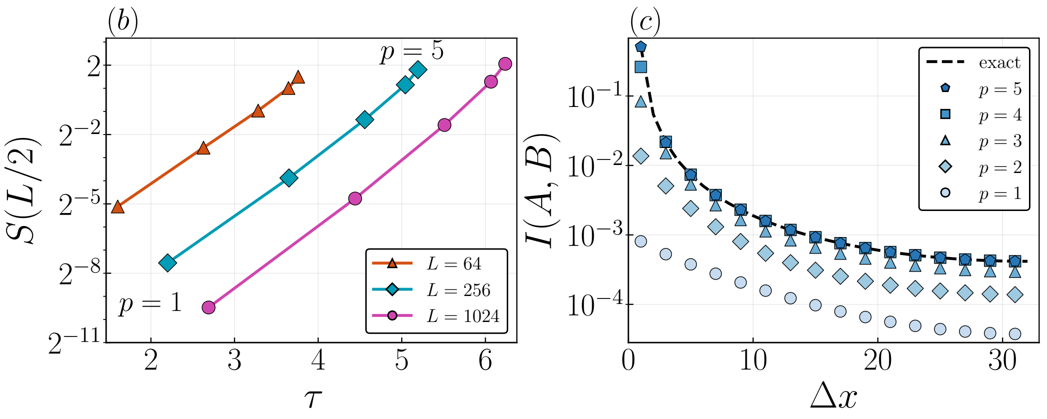


Figure 5.9: (a) Entanglement entropy of half partition grows exponentially with imaginary time τ, in the optimal $P = 5$ ansatz for the critical state. (b) Mutual information between two spins at positions A and B respectively as a function of their distance Δx, for intermediate steps p in the $P = 5$ protocol with $L = 64$.

on system size L, field h although steps from a fixed P to a different P cause sudden changes in the ideal parameters.

One reason for the success of the ansatz is that each step p contributes an exponential amount of entanglement. This reveals why it is more powerful than the real-time QAOA which has linear scaling, $\tau \sim L$.

In the next Section, we use this ansatz in the framework of VMC.

5.4 Variational Monte Carlo

While the TFIM model admits a dual representation as free fermions, for a general model sampling methods are crucial for estimating the energy cost function. As a proof of concept, we also use Monte Carlo sampling for stochastically optimizing VITA.

The quantum-classical correspondence maps quantum observables to dual classical observables of a classical anisotropic Ising model on an $L \times (2P + 1)$ lattice. We denote the classical spin configurations by $\{s\}$ and the spatial and imaginary time by (i, p), respectively[2].

[2]This is a slight abuse of notation since $1 \leq p \leq P$ in Eq. (5.3). However, this may be permitted since there are only p unique time couplings $J_t(p)$ due to the lattice symmetry.

The expectation value of a quantum observable $\mathcal{O}$ is

$$\langle\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})|\mathcal{O}|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle = \sum_{\{s\}} \tilde{\mathcal{O}}(s)\, p_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s) \tag{5.35}$$

where $\tilde{\mathcal{O}}$ are dual classical observables, and $p_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s)$ is the Boltzmann weight corresponding to the Ising model with couplings $J_x(p) = \alpha_p$, $J_\tau(p) = \frac{1}{2}\ln\coth\beta_p$ between nearest-neighbours in space and imaginary time, respectively. The couplings vary with imaginary time, p, but are uniform in space. In this way, the energy of the trial wavefunction can be sampled efficiently with Monte Carlo.

5.4.1 Quantum to classical mapping

The foundation of path integral world-line Monte Carlo is the connection between a quantum system in d-dimensions and a classical statistical one in $(d+1)$-dimensions. In our case, we wish to consider the variational quantum state given by

$$|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle = \mathcal{N}\prod_{p=P}^{1} e^{-\beta_p H_X} e^{-\alpha_p H_{ZZ}}\,|+\rangle\ . \tag{5.36}$$

In the following, we will only consider $P = 1$, but the higher-P formulation is nearly identical. The expectation value of some operator $\mathcal{O}$, up to a normalization, is

$$\begin{aligned}\langle\mathcal{O}\rangle &= \langle\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})|\,\mathcal{O}\,|\psi_P(\boldsymbol{\alpha},\boldsymbol{\beta})\rangle\\ &= \langle+|e^{-\alpha_1 H_{ZZ}}e^{-\beta_1 H_X}\mathcal{O}e^{-\beta_1 H_X}e^{-\alpha_1 H_{ZZ}}|+\rangle\\ &= \sum_{\{s\}}\sum_{i=1}^{L}\sum_{p=1}^{2P+1}\tilde{\mathcal{O}}(s_{i,p})\,p(s_{i,p})\,. \end{aligned}\tag{5.37}$$

The probability $p(s_{i,p})$ is the Boltzmann factor, given by the exponential of the Hamiltonian

$$H_{2\mathrm{D}} = -\sum_{i=1}^{L}\sum_{p=1}^{2P+1}\left(J_x(p)s_{i,p}s_{i+1,p} + J_t(p)s_{i,p}s_{i,p+1}\right) \tag{5.38}$$

up to a normalization $Z = \sum e^{-H_{2\text{D}}}$, where the couplings of the classical model are related to $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ via

$$J_x(p) = \alpha_p \tag{5.39}$$

$$J_t(p) = \frac{1}{2} \ln \coth \beta_p \tag{5.40}$$

Continuing this process for higher P is straightforward, and can be visualized as a $L \times (2P+1)$ lattice in Fig. 5.10. Adding connections $J_x(0)$ between the middles time-slices $(p = P+1)$ one obtains the Suzuki-Trotter inspired neural networks in Ref. [177].

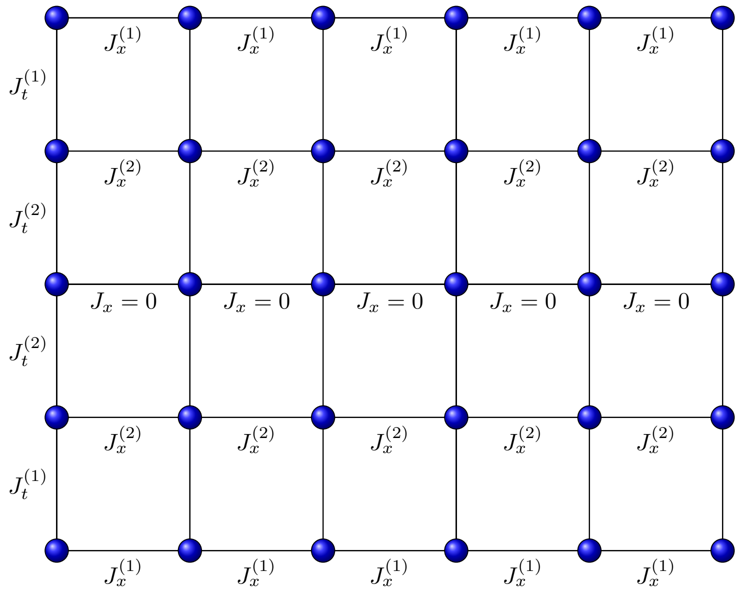


Figure 5.10: Schematic of the two-dimensional classical Ising lattice dual to the ansatz in Eq. (5.36) for depth $P = 2$ and $L = 6$ spins.

The only observables required to compute the total energy are:

$$\langle Z_i Z_{i+1} \rangle = \sum_{\{\mathbf{s}\}} s_{i,m}\, s_{i,m}\, p(s_{i,p}) \tag{5.41}$$

$$\langle X_i \rangle = \sum_{\{\mathbf{s}\}} \exp(-2J_t(m)\, s_{i,m}, s_{i,m}) p(s_{i,p}) \tag{5.42}$$

$$\langle Z_i \rangle = \sum_{\{\mathbf{s}\}} s_{i,m} p\left(s_{i,p}\right) \tag{5.43}$$

where $m = P + 1$ denotes the middle time-slice.

Gradients of these expressions can be evaluated via

$$\partial_\theta \langle H \rangle = \langle \partial_\theta H \rangle - \langle H \partial_\theta E_{\boldsymbol{\alpha},\boldsymbol{\beta}} \rangle + \langle E_{\boldsymbol{\alpha},\boldsymbol{\beta}} \rangle \langle H \rangle \tag{5.44}$$

where $E_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ is the energy of the classical Ising model, and $\boldsymbol{\theta}$ denotes $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. The exponential form of Eq. (5.36) make stochastic reconfiguration (SR) [12], easy to implement. SR uses the positive-definite covariance matrix S as a pre-conditioner for the gradient where S is defined as

$$S_{\theta,\theta'} = \langle \mathcal{O}_\theta \mathcal{O}_{\theta'} \rangle - \langle \mathcal{O}_\theta \rangle \langle \mathcal{O}_{\theta'} \rangle \ , \tag{5.45}$$

with $\mathcal{O}_\theta = \partial_\theta(\log \psi(\boldsymbol{\theta}))$.

Each gradient update then takes the form

$$\theta \to \theta - \eta S^{-1} \partial_\theta \langle H \rangle \tag{5.46}$$

for a small learning rate η which we take to decay during iteration. It is possible that S is non-invertible, so we use the Moore-Penrose pseudo-inverse.

5.4.2 Results

We use this scheme with $P = 1, 2$ to target the ground states for various values of the transverse field h in both the (integrable) one-dimensional and the (non-integrable) two-dimensional TFIM. Stochastic natural gradient descent (stochastic reconfiguration) is used to optimize the parameters [180]. We find rapid convergence for $P = 1$, while higher-P becomes more difficult, especially for with a noisy objective function.

The relative error in energy achieved is shown in Fig. 5.11, with the free fermion results for comparison. For $P = 1$ the VMC achieves the same accuracy as the free fermion method. However, for $P = 2$ away from the critical point, the VMC performance is limited by sampling error[3].

We also test the model on the two-dimensional TFIM for a 10×10 lattice. The reference energy is computed using zero-temperature stochastic series expansion [221, 222, 223]. In

[3]Of course the statistical error in Monte Carlo can be made arbitrarily small given long enough run-time since the sampling error goes as $\mathcal{O}(N_{\mathrm{MC}}^{-1/2})$ for N_{MC} Monte Carlo sweeps. This guarantees that VMC will converge to the free fermion solution in Fig. 5.11a if the globally optimal parameters can be found. The computational time of the MC sampling is $\mathcal{O}(\tau_{\mathrm{corr}} N_{\mathrm{MC}})$ where the autocorrelation time $\tau_{\mathrm{corr}} \sim (PN)^\gamma$ is a polynomial function of PN for N spins and P pulses of VITA.

two-dimensions, we again find the $P = 1$ ansatz very efficient. However, increasing P in this case did not result is a significant improvement in energy. This is likely due noisy gradients during optimization.

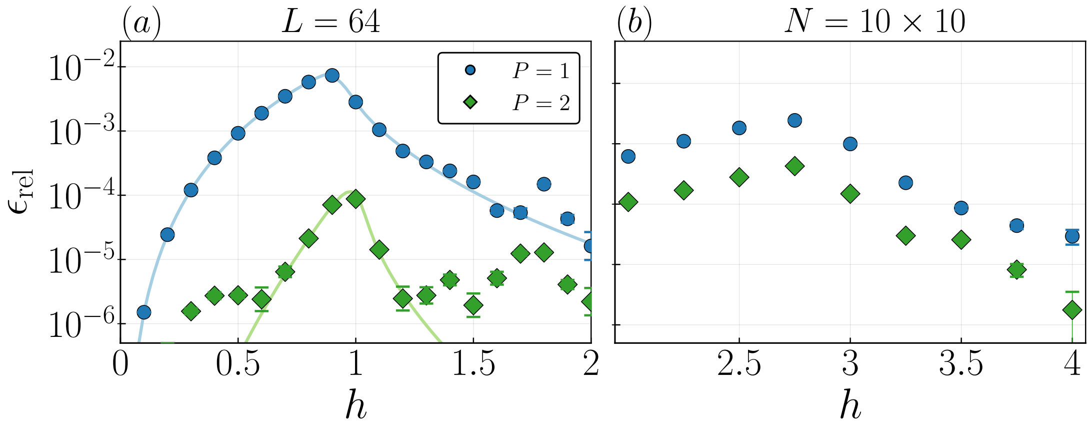


Figure 5.11: Relative error in energy, ϵ_{rel} for VMC using our ansatz on the TFIM: (a) 1d model with $L = 64$ spins. Solid lines denotes the results from the free fermion approach, (b) 2d model on a 10×10 square lattice. Energies are compared with those from zero-temperature stochastic series expansion.

Non-integrable longitudinal model

As an extension, we consider adding a longitudinal field $-h_Z \sum_i Z_i$ to the critical Hamiltonian ($h/J = 1$) so that the model is non-integrable. The Hamiltonian is

$$H = -J \sum_i Z_i Z_{i+1} - h \sum_i X_i - h_Z \sum_i Z_i \,. \tag{5.47}$$

This model has no closed-form solution, however DMRG rapidly find the ground state for a one-dimensional system. This provides a benchmark to compare with our ansatz.

To accommodate the extra term in the Hamiltonian, the VITA also needs an additional parameter γ_p imposed by changing $\alpha_p H_{ZZ} \to \alpha_p H_{ZZ} + \gamma_p H_Z$ in Eq. (5.3) so the ansatz is

$$|\psi_P(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)\rangle = \mathcal{N} \prod_{p=P}^{1} e^{-\beta_p H_X} e^{-\alpha_p H_{Zz} - \gamma_p H_Z} |+\rangle \,. \tag{5.48}$$

In the classical-to-quantum mapping, the additional parameters γ_p becomes an on-site field term at each lattice point. Accordingly, it remains efficient to sample with Metropolis-Hastings or cluster algorithms. In Fig. 5.12, we see that VITA with $P = 1$ continues to yield accurate energies in the enlarged phase diagram. In fact, the least accurate point is the critical point $h = 1, h_Z = 0$.

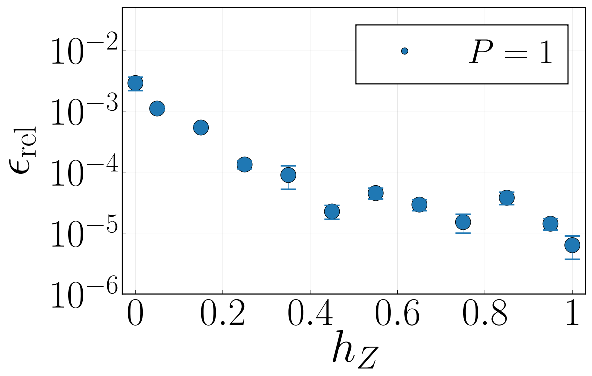


Figure 5.12: Relative error in energy, $\epsilon_{\rm rel}$ for VMC using our ansatz on the TFIM with a longitudinal field from Eq. 5.47.

5.5 Conclusion

We have introduced a variational technique that is motivated by both projector methods and recently developed quantum algorithms. It provides substantial shortcuts to the usual trotterization of imaginary time evolution, at the expense of making the procedure variational. Using TFIM as a first test bed, we have demonstrated that this ansatz is viable for sampling methods and highly efficient. In particular, the number of variational parameters required to represent the TFIM critical state scales as $\sim \log L$, in contrast to other variational methods such as density matrix renormalization group (DMRG), which in this critical case requires bond dimension scaling with L and thus number of parameters scaling with L^2. One reason for this efficiency is the fact that imaginary time evolution is not subject to many bounds for real time evolution; despite being generated by local Hamiltonians, our ansatz exhibits an exponential growth of entanglement entropy and rapid generation of long-range correlations, features unique to imaginary time evolution.

Our variational approach is potentially useful in the many situations where imaginary time trotterization involves a prohibitively large number of steps. For example, in models with a sign problem, the computational cost scales exponentially with space and imaginary time $\mathcal{O}(\tau L^d)$. Our ansatz provides a variational shortcut that significantly reduces τ (from $\tau \sim L$ to $\tau \sim (\log L)^{2.3}$ in the critical 1d TFIM) which could enable the study of larger systems even with a sign problem. For few variational parameters ($P = 1, 2$), the optimization may be feasible, and we leave these investigations to future work.

In the case of the two-dimensional Hubbard model, the sign problem occurs away from half-filling. VITA in this case is a generalization of the Gutzwiller variational wavefunction. By alternating between the two non-commuting Hamiltonians, VITA is a variational version of auxiliary-field quantum Monte Carlo with few Trotter steps.[4] The trade-off is that larger steps are needed. This will lead to stability issues. In the usual auxiliary-field determinant Monte Carlo, stabilization is needed after a few Trotter steps. In the case of VITA, it is needed after every (large) step.

In a sense, VITA is a hybrid variational-worldline Monte Carlo method. In the limit of many parameters, VITA reduces to the usual discrete-time Trotter formulation of quantum Monte Carlo. Other worldline algorithms such as the Stochastic Series Expansion [29] can be recast into the VITA form by having time-dependant couplings. The modern software infrastructure of deep learning, namely automatic differentiation provides a way to adapt current quantum Monte Carlo algorithms into the VITA variational form with little overhead.

[4]For a nice review of world-line Monte Carlo methods see Assaad and Evertz [28].

6

Discussion and Outlook

For better or worse we are now witnessing a transition from the science of the past, so intimately linked to reductionism, to the study of complex adaptive matter, firmly based in experiment, with its hope for providing a jumping-off point for new discoveries, new concepts, and new wisdom.

– Laughlin and Pines, *The Theory of Everything*, 2000

The cycle of science has often preceded in a symbiotic relationship alternating between experimental realizations and theoretical predictions. Wildly successful machine learning algorithms offer a new catalyst to accelerate this cycle. By leveraging emergent properties in complex neural networks, we are able to identify patterns and statistical features from large data sets. In many practical problems, such those in natural language, computer vision, and games of strategy, neural networks have provided a heuristic way to tame the curse of dimensionality. It is the purpose of this Thesis to explore the application of neural networks to the high-dimensional problems present in many-body physics.

Strongly interacting systems poses an exponentially hard problem for classical computing. Physical insight is key to make advances in these problems. While physicists have relied on traditional numerical or analytical methods to gain insights, with machine learning we gain knowledge from a very different perspective. In this Thesis, we have investigated many facets of this problem. We studied the interpretability of algorithms, connections with renormalization, quantum-inspired variational models, and tools for experimental design.

Topological defects: Classification, bias, and interpretability

As a first investigation, in Chapter 2 we used standard neural networks to classify topological phases in the classical XY model. Even though a convolutional network is capable of using vortices as an internal representation, we found that it is not practical, nor even beneficial to do so for small system sizes. From this we gained valuable insights. Firstly, neural networks tend to follow the path of least resistance to distinguish of phases; in this case, a local quantity like the magnetization. Secondly, when considering finite-size samples, we need to be extremely careful; while physicists may be interested in the thermodynamic limit, a neural network is not.

This is relevant to the theme of fairness and equality in ML research. In many data sets, unwanted social biases enter in the form of under-respresentation such as race and gender [224, 225, 226]. One promising idea is "learning not to learn" by Kim et al. [227] which explicitly penalizes learning biases. We can use this XY model to prevent learning magnetization, or energy, but since it is an *infinite*-order transition, there is always some quantity that distinguishes the phases.

This Chapter also presented the need for interpretable neural networks. Further work in this direction by Wetzel and Scherzer [228] indicated that looking at network decision functions provided an explicit way to find out what a classification network is learning. Other examples include learning invariants such as Lorentz transformations [229] and winding numbers [230, 231, 232].

During the development of this project, we implemented rotationally invariant convolution filters to incorporate the symmetry of vortices. Enforcing symmetries is typical in physics and is becoming more common place in ML research [233, 234, 235, 236]. For example, group equivariant convolutional filters proposed by Cohen et al. [235] generalize convolutions to possess discrete group symmetries including translations, reflections and rotations. This directly addresses the efficiency of neural networks.

Super-resolution and renormalization

The connections between renormalization and deep learning has been of constant interest to both communities [105, 95, 107, 106]. To extend this, we used super-resolution to invert the renormalization group decimation for the Ising model in Chapter 3. We first applied super-resolution to the one-dimension Ising model. Super-resolution works very well for this model since it is self-similar under renormalization. Further, extrapolating to much larger systems yields consistent results as measured by the two-point function. More ambitiously, we used super-resolution for the two-dimensional model where it is not self-similar. In this case, the error in upscaling increases as the extrapolation size increases, in agreement with intuition.

This Chapter was the first demonstration of using super-resolution on an ensemble opposed to single-image cases. This allows us judge to performance of the super-resolving network based on statistical averages instead compared to single-image metrics. Similar to learning topological defects in Chapter 2, this provides novel insight into what the neural network is learning when upscaling. The key difficultly is underestimation of the number of blocks with an equal number of up and down spins. This results in an error that increases as the system size increases.

Using super-resolution provides a way of choosing high-probability states for initialization in Monte Carlo, hence reducing the warm-up time as suggested by Hastings [37]. The only expense is having to first simulate a smaller system and perform a decimation in order to train the network. Similar coarse-graining techniques are used in lattice quantum field theory where we expect this technique could be useful [237].

In our case, we approximated the rescaling transformation with the manual curve-fitting procedure. However, this could also be automated by using a neural network. This improvement would generalize this method to much more complex Hamiltonian's with more interacting terms. One example could be classical spin glasses, where initializing each Monte Carlo chain near a local energy minima of a smaller system could result in faster convergence.

Lastly, with only one system size and a decimation procedure, we were able to estimate critical exponents. This could be a powerful technique for more difficult interacting systems where large-scale Monte Carlo simulation is prohibitively expensive.

Calibrating quantum devices with wavefunction reconstruction

Monte Carlo provides a way to generate snapshots of microscopic configurations of classical or quantum data. However, ultimately physics is an experimental science where we can generate data from measurements. One recent source of interesting measurements is in modern quantum devices. Projective spin measurements can be measured for a quantum circuit and used as data to train a neural network model.

In Chapter 4, we used restricted Boltzmann machines and self-attention transformer networks for performing maximum-likelihood quantum state tomography. RBMs offer the advantage of a network architecture inspired by statistical physics. In contrast, transformers provide greater accuracy at the expense of interpretability. In both cases the networks were able to accurately capture the relevant physics when trained on projective spin measurements for the quantum Ising model.

The power of generative models lies in the ability to estimate off-diagonal observables that are inaccessible in both experiment and Monte Carlo simulation. These models could also likely be leveraged for variation Monte Carlo since they produce explicit probabilities for each configuration. Recently, explicit density estimation models such as transformers and recurrent networks have found use in variational Monte Carlo [238, 41, 239, 240].

With the growing experimental realization of quantum devices, it is important to calibrate these devices accordingly. Machine learning methods have reduced the bottleneck from maximum-likelihood tomography. This will enable more robust testing of quantum devices that may eventually be used for machine learning. This feedback loop could create better machine learning models which could be in turn used to the design of better quantum computers. In a sense, the most appropriate machine learning for quantum devices might be quantum machine learning [241].

Quantum-inspired variational Monte Carlo

One of the most promising applications of quantum computers is to solve combinatorial optimization problems. A recent algorithm, the quantum approximate optimization algorithm (QAOA) is a leading candidate. In this Thesis, we studied a quantum-inspired variational ansatz. Whereas QAOA is inspired by discrete version of the quantum adiabatic algorithm, the variational imaginary time ansatz (VITA) is a hybrid between variational and path integral Monte Carlo techniques. One can also consider it a latent variable machine learning model where the latent space is physically motivated by the mapping of a d-dimensional quantum system to a $(d+1)$-dimensional classical system.

For the quantum Ising model, we found logarithmic scaling of the total imaginary time with system size. This suggests the variational model could be useful when very long imaginary-times are needed in traditional quantum Monte Carlo. Our result is also an exponential improvement over the real-time scaling in QAOA. Since imaginary time can be efficiently simulated, this suggests that new algorithms for quantum devices need to be discovered to achieve a quantum advantage over classical methods.

The power of the classical algorithm may be due to the exponential growth of entanglement in each layer of the ansatz, in contrast to the quantum algorithm that is subject to bounded entanglement growth. A natural extension of this model is to apply the VITA to sign-problematic models such as the two-dimensional Hubbard model. The usual auxiliary-field projector quantum Monte Carlo (PQMC) algorithm [28] used to study the Hubbard model can be adapted to use variational Trotter steps. In this way, we can start a simulation with state-of-the-art auxiliary-field Monte Carlo and fine-tune with adjustable parameters. The parameters could be optimized with gradient-based algorithms using either automatic differentiation or analytic derivatives (from higher-order correlation functions).

Similarly, other algorithms such as stochastic series expansion [29] or variational tensor networks [175] could be modified within the VITA framework. Matrix product state tensor networks could be promising since they use more efficient autoregressive sampling compared to Monte Carlo sampling.

The improvement of classical algorithms is important for finding the division between classical and quantum computation. Quantum-inspired methods could improve current hybrid quantum-classical algorithms [242, 243, 244, 245] where the bottleneck is accessing quantum resources.

Summary & Outlook

When faced with the dual obstacles of scale and complexity, many interesting problems cannot be solved from only fundamental laws. It is precisely the interactions and higher-organizing principles in nature that give matter its unique and interesting properties. Since the 1860s, physicists have studied the emergence of large-scale macroscopic properties from their microscopic constituents. Machine learning promises a radically new approach. We are shifting from studying emergence in matter, to designing systems that use emergence to solve otherwise impossible tasks.

References

[1] Matthew J. S. Beach, Anna Golubeva, and Roger G. Melko. Machine learning vortices at the Kosterlitz-Thouless transition. *Physical Review B*, 97(4):045207, January 2018.

[2] Matthew J. S. Beach. Machine Learning Topological Defects in the XY Model — ⟨ physics — machine learning ⟩.

[3] Stavros Efthymiou, Matthew J. S. Beach, and Roger G. Melko. Super-resolving the Ising model with convolutional neural networks. *Physical Review B*, 99(7):075113, February 2019.

[4] Matthew J. S. Beach, Roger G. Melko, Tarun Grover, and Timothy H. Hsieh. Making trotters sprint: A variational imaginary time ansatz for quantum many-body systems. *Physical Review B*, 100(9):094434, September 2019.

[5] Matthew J. S. Beach, Isaac De Vlugt, Anna Golubeva, Patrick Huembeli, Bohdan Kulchytskyy, Xiuzhe Luo, Roger Melko, Ejaaz Merali, and Giacomo Torlai. QuCumber: Wavefunction reconstruction with neural networks. *SciPost Physics*, 7(1):009, July 2019.

[6] W. Kohn. An essay on condensed matter physics in the twentieth century. *Reviews of Modern Physics*, 71(2):S59–S77, March 1999.

[7] Elbio Dagotto. Correlated electrons in high-temperature superconductors. *Reviews of Modern Physics*, 66(3):763–840, July 1994.

[8] Rahul Nandkishore and David A. Huse. Many-Body Localization and Thermalization in Quantum Statistical Mechanics. *Annual Review of Condensed Matter Physics*, 6(1):15–38, 2015. _eprint: https://doi.org/10.1146/annurev-conmatphys-031214-014726.

[9] P. W. Anderson. More Is Different. *Science*, 177(4047):393–396, August 1972.

[10] R. B. Laughlin and David Pines. The Theory of Everything. *Proceedings of the National Academy of Sciences of the United States of America*, 97(1):28–31, January 2000.

[11] Roman Orus. A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. *Annals of Physics*, 349:117–158, October 2014.

[12] Federico Becca and Sandro Sorella. *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, Cambridge, United Kingdom ; New York, NY, 1 edition edition, November 2017.

[13] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602, February 2017.

[14] H. Bethe. Zur Theorie der Metalle. *Zeitschrift für Physik*, 71(3):205–226, March 1931.

[15] Michael Karbach and Gerhard Muller. Introduction to the Bethe ansatz I. *arXiv:cond-mat/9809162*, September 1998.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[17] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. NVIDIA Tensor Core Programmability, Performance & Precision. *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 522–531, May 2018.

[18] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg,

Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. *arXiv:1704.04760 [cs]*, April 2017.

[19] Phelim P. Boyle. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4(3):323–338, May 1977.

[20] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. New York : Springer, 2004.

[21] Charles J. Mode. *Applications of Monte Carlo Methods in Biology, Medicine and Other Fields of Science*. February 2011.

[22] Stefan Weinzierl. Introduction to Monte Carlo methods. June 2000.

[23] Washington Taylor and Yi-Nan Wang. A Monte Carlo exploration of threefold base geometries for 4d F-theory vacua. October 2015.

[24] K. P. N. Murthy. An Introduction to Monte Carlo Simulation of Statistical physics Problem. *arXiv:cond-mat/0104167*, December 2003.

[25] Jean-Charles Walter and Gerard Barkema. An introduction to Monte Carlo methods. *Physica A: Statistical Mechanics and its Applications*, 418:78–87, January 2015.

[26] K. Binder and Steven M. Girvin. The Monte Carlo method in condensed matter physics. 1992.

[27] Anders W. Sandvik. Computational Studies of Quantum Spin Systems. In *AIP Conference Proceedings*, volume 1297, pages 135–338, November 2010.

[28] F.F. Assaad and H.G. Evertz. World-line and Determinantal Quantum Monte Carlo Methods for Spins, Phonons and Electrons. In H. Fehske, R. Schneider, and A. Weiße, editors, *Computational Many-Particle Physics*, Lecture Notes in Physics, pages 277–356. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[29] Anders W. Sandvik. Stochastic series expansion method with operator-loop update. *Physical Review B*, 59(22):R14157–R14160, June 1999.

[30] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar. Monte Carlo calculations of coupled boson-fermion systems. I. *Physical Review D*, 24(8):2278–2286, October 1981.

[31] W. L. McMillan. Ground State of Liquid ${\mathrm{He}}\hat{4}$. *Physical Review*, 138(2A):A442–A451, April 1965.

[32] D. Ceperley, G. V. Chester, and M. H. Kalos. Monte Carlo simulation of a many-fermion study. *Physical Review B*, 16(7):3081–3099, October 1977.

[33] M. P. Nightingale and H. W. J. Blöte. Dynamic Exponent of the Two-Dimensional Ising Model and Monte Carlo Computation of the Subdominant Eigenvalue of the Stochastic Matrix. *Physical Review Letters*, 76(24):4548–4551, June 1996.

[34] Robert Swendsen and Jian-Sheng Wang. Replica Monte Carlo Simulation of Spin-Glasses. *Physical review letters*, 57:2607–2609, December 1986.

[35] Koji Hukushima and Koji Nemoto. Exchange Monte Carlo Method and Application to Spin Glass Simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, June 1996.

[36] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[37] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

[38] Robert H. Swendsen. Monte Carlo calculation of renormalized coupling parameters. I. d=2 Ising model. *Physical Review B*, 30(7):3866–3874, October 1984.

[39] Ulli Wolff. Collective Monte Carlo Updating for Spin Systems. *Physical Review Letters*, 62(4):361–364, January 1989.

[40] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, Lecture Notes in Computer Science, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[41] Andrew J. Ferris and Guifre Vidal. Perfect sampling with unitary tensor networks. *Physical Review B*, 85(16):165146, April 2012.

[42] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019.

[43] Harshit Sharma. Auto-Regressive Generative Models (PixelRNN, PixelCNN++), December 2017.

[44] Bohdan Kulchytskyy. *Probing Universality with Entanglement Entropy via Quantum Monte Carlo*. PhD thesis, University of Waterloo, Waterloo, ON, August 2019. Accepted: 2019-08-30T19:41:28Z.

[45] Lauren Hayward Sierens. *Simulating Quantum Matter through Lattice Field Theories*. PhD thesis, University of Waterloo, Waterloo, ON, May 2017.

[46] Andrej Karpathy. Software 2.0 - Andrej Karpathy, June 2018.

[47] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv:1909.08053 [cs]*, March 2020.

[48] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, May 2020.

[49] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019.

[50] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore

Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[51] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1962.

[52] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, January 1991.

[53] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, January 1993.

[54] G Cybenkot. Approximation by superpositions of a sigmoidal function. page 12.

[55] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation,.* Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2008.

[56] Atilim Gunes Baydin and Barak A. Pearlmutter. Automatic Differentiation of Algorithms for Machine Learning. *arXiv:1404.7456 [cs, stat]*, April 2014.

[57] Henry W. Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, September 2017.

[58] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12(61):2121–2159, July 2011.

[59] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014.

[60] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, May 2017.

[61] Hai-Jun Liao, Jin-Guo Liu, Lei Wang, and Tao Xiang. Differentiable Programming Tensor Networks. *Physical Review X*, 9(3):031041, September 2019.

[62] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design. *arXiv:1912.10877 [cond-mat, physics:physics, physics:quant-ph]*, December 2019.

[63] Teresa Tamayo-Mendoza, Christoph Kreisbeck, Roland Lindh, and Alán Aspuru-Guzik. Automatic Differentiation in Quantum Chemistry with Applications to Fully Variational Hartree–Fock. *ACS Central Science*, 4(5):559–566, May 2018.

[64] V. L. Berezinskiǐ. Destruction of Long-range Order in One-dimensional and Two-dimensional Systems having a Continuous Symmetry Group I. Classical Systems. *Soviet Journal of Experimental and Theoretical Physics*, 32:493, 1971.

[65] V. L. Berezinskiǐ. Destruction of Long-range Order in One-dimensional and Two-dimensional Systems Possessing a Continuous Symmetry Group. II. Quantum Systems. *Soviet Journal of Experimental and Theoretical Physics*, 34:610, 1972.

[66] J. M. Kosterlitz and D. J. Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C: Solid State Physics*, 6(7):1181–1203, April 1973.

[67] P. Kapitza. Viscosity of Liquid Helium below the λ-Point. *Nature*, 141(3558):74–74, January 1938.

[68] J. F. Allen and A. D. Misener. Flow of Liquid Helium II. *Nature*, 141(3558):75–75, January 1938.

[69] D. J. Bishop and J. D. Reppy. Study of the Superfluid Transition in Two-Dimensional $\hat{\{}4}\mathrm{He}$ Films. *Physical Review Letters*, 40(26):1727–1730, June 1978.

[70] Sebastian J. Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Physical Review E*, 96(2):022140, August 2017.

[71] Wenjian Hu, Rajiv R. P. Singh, and Richard T. Scalettar. Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination. *Physical Review E*, 95(6):062122, June 2017.

[72] Lei Wang. Discovering phase transitions with unsupervised learning. *Physical Review B*, 94(19):195105, November 2016.

[73] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014.

[74] Marco Cristoforetti, Giuseppe Jurman, Andrea I. Nardelli, and Cesare Furlanello. Towards meaningful physics from generative models. *arXiv:1705.09524 [cond-mat, physics:hep-lat]*, May 2017.

[75] Peter Broecker, Fakher F. Assaad, and Simon Trebst. Quantum phase recognition via unsupervised machine learning. *arXiv:1707.00663 [cond-mat]*, July 2017.

[76] Pengfei Zhang, Huitao Shen, and Hui Zhai. Machine Learning Topological Invariants with Neural Networks. *Physical Review Letters*, 120(6):066401, February 2018.

[77] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, May 2017.

[78] N. D. Mermin and H. Wagner. Absence of Ferromagnetism or Antiferromagnetism in One- or Two-Dimensional Isotropic Heisenberg Models. *Physical Review Letters*, 17(22):1133–1136, November 1966.

[79] S. G. Chung. Essential finite-size effect in the two-dimensional XY model. *Physical Review B*, 60(16):11761–11764, October 1999.

[80] Rudolf Podgornik. Principles of condensed matter physics. P. M. Chaikin and T. C. Lubensky, Cambridge University Press, Cambridge, England, 1995. *Journal of Statistical Physics*, 83:1263–1265, June 1996.

[81] Peter Olsson. Monte Carlo analysis of the two-dimensional XY model. II. Comparison with the Kosterlitz renormalization-group equations. *Physical Review B*, 52(6):4526–4535, August 1995.

[82] David R. Nelson and J. M. Kosterlitz. Universal Jump in the Superfluid Density of Two-Dimensional Superfluids. *Physical Review Letters*, 39(19):1201–1205, November 1977.

[83] Petter Minnhagen. The two-dimensional Coulomb gas, vortex unbinding, and superfluid-superconducting films. *Reviews of Modern Physics*, 59(4):1001–1066, October 1987.

[84] Lutz Prechelt. Early Stopping — But When? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, Lecture Notes in Computer Science, pages 53–67. Springer, Berlin, Heidelberg, 2012.

[85] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and

Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16*, pages 265–283, 2016.

[86] Francois Chollet. Keras, 2015.

[87] Pedro Ponte and Roger G. Melko. Kernel methods for interpretable machine learning of order parameters. *Physical Review B*, 96(20):205146, November 2017.

[88] Yuki Nagai, Huitao Shen, Yang Qi, Junwei Liu, and Liang Fu. Self-learning Monte Carlo method: Continuous-time algorithm. *Physical Review B*, 96(16):161102, October 2017.

[89] Ce Wang and Hui Zhai. Machine learning of frustrated classical spin models. I. Principal component analysis. *Physical Review B*, 96(14):144432, October 2017.

[90] S. T. Bramwell and P. C. W. Holdsworth. Magnetization: A characteristic of the Kosterlitz-Thouless-Berezinskii transition. *Physical Review B*, 49(13):8811–8814, April 1994.

[91] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[92] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, April 2016.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[94] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 818–833, Cham, 2014. Springer International Publishing.

[95] Pankaj Mehta and David J. Schwab. An exact mapping between the Variational Renormalization Group and Deep Learning. October 2014.

[96] S. R. Manmana, S. Wessel, R. M. Noack, and A. Muramatsu. Strongly Correlated Fermions after a Quantum Quench. *Physical Review Letters*, 98(21):210405, May 2007.

[97] Giacomo Torlai and Roger G. Melko. Learning thermodynamics with Boltzmann machines. *Physical Review B*, 94(16):165134, October 2016.

[98] Zhaocheng Liu, Sean P. Rodrigues, and Wenshan Cai. Simulating the Ising Model with a Deep Convolutional Generative Adversarial Network. *arXiv:1710.04987 [cond-mat]*, October 2017.

[99] Giuseppe Carleo, Yusuke Nomura, and Masatoshi Imada. Constructing exact representations of quantum many-body systems with deep neural networks. *Nature Communications*, 9(1), December 2018.

[100] E. M. Inack, G. E. Santoro, L. Dell'Anna, and S. Pilati. Projective quantum Monte Carlo simulations guided by unrestricted neural network states. *Physical Review B*, 98(23):235145, December 2018.

[101] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447, May 2018.

[102] Giacomo Torlai and Roger G. Melko. Latent Space Purification via Neural Density Operators. *Physical Review Letters*, 120(24):240503, June 2018.

[103] John Cardy. Scaling and Renormalization in Statistical Physics. *Scaling and Renormalization in Statistical Physics, by John Cardy, Cambridge, UK: Cambridge University Press, 1996*, May 1996.

[104] Leo Kadanoff. *Statistical Physics: Statics, Dynamics and Renormalization*. World Scientific Publishing Company, Singapore ; River Edge, N.J, May 2000.

[105] Cédric Bény. Deep learning and the renormalization group. *arXiv:1301.3124 [quant-ph]*, January 2013.

[106] Shuo-Hui Li and Lei Wang. Neural Network Renormalization Group. *Physical Review Letters*, 121(26):260601, December 2018.

[107] Maciej Koch-Janusz and Zohar Ringel. Mutual Information, Neural Networks and the Renormalization Group. *Nature Physics*, 14(6):578–582, June 2018.

[108] Patrick M. Lenggenhager, Doruk Efe Gökmen, Zohar Ringel, Sebastian D. Huber, and Maciej Koch-Janusz. Optimal Renormalization Group Transformation from Information Theory. *arXiv:1809.09632 [cond-mat]*, October 2019.

[109] Satoshi Iso, Shotaro Shiba, and Sumito Yokoo. Scale-invariant feature extraction of neural network and renormalization group flow. *Physical Review E*, 97(5):053304, May 2018.

[110] Kyle Mills, Kevin Ryczko, Iryna Luchak, Adam Domurad, Chris Beeler, and Isaac Tamblyn. Extensive deep neural networks for transferring small scale learning to large scale systems. *Chemical Science*, 10(15):4129–4140, 2019.

[111] C. Dong, C. C. Loy, K. He, and X. Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, February 2016.

[112] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv:1609.04802 [cs, stat]*, May 2017.

[113] Dorit Ron, Robert H. Swendsen, and Achi Brandt. Inverse Monte Carlo Renormalization Group Transformations for Critical Phenomena. *Physical Review Letters*, 89(27):275701, December 2002.

[114] Mehran Kardar. *Statistical Physics of Fields*. Cambridge University Press, Cambridge; New York, 1 edition edition, June 2007.

[115] Rodney J. Baxter. *Exactly Solved Models in Statistical Mechanics*. DOv.

[116] Robert H. Swendsen. Monte Carlo Renormalization Group. *Physical Review Letters*, 42(14):859–861, April 1979.

[117] Shang-keng Ma, Chandan Dasgupta, and Chin-kun Hu. Random Antiferromagnetic Chain. *Physical Review Letters*, 43(19):1434–1437, November 1979.

[118] Chandan Dasgupta and Shang-keng Ma. Low-temperature properties of the random Heisenberg antiferromagnetic chain. *Physical Review B*, 22(3):1305–1319, August 1980.

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

[120] Nikolaj Moll, Panagiotis Barkoutsos, Lev S. Bishop, Jerry M. Chow, Andrew Cross, Daniel J. Egger, Stefan Filipp, Andreas Fuhrer, Jay M. Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

[121] Hannes Bernien, Sylvain Schwartz, Alexander Keesling, Harry Levine, Ahmed Omran, Hannes Pichler, Soonwon Choi, Alexander S. Zibrov, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551(7682):579–584, November 2017.

[122] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe. Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator. *Nature*, 551(7682):601–604, November 2017.

[123] S. R. White, R. L. Sugar, and R. T. Scalettar. Algorithm for the simulation of many-electron systems at low temperatures. *Physical Review B*, 38(16):11665–11668, December 1988.

[124] Ming-Chiang Chung and Ingo Peschel. Density-matrix spectra of solvable fermionic systems. *Physical Review B*, 64(6):064412, July 2001.

[125] Guifré Vidal. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Physical Review Letters*, 91(14):147902, October 2003.

[126] Marcus Cramer, Martin B. Plenio, Steven T. Flammia, Rolando Somma, David Gross, Stephen D. Bartlett, Olivier Landon-Cardinal, David Poulin, and Yi-Kai Liu. Efficient quantum state tomography. *Nature Communications*, 1(1):1–7, December 2010.

[127] B. P. Lanyon, C. Maier, M. Holzäpfel, T. Baumgratz, C. Hempel, P. Jurcevic, I. Dhand, A. S. Buyskikh, A. J. Daley, M. Cramer, M. B. Plenio, R. Blatt, and C. F. Roos. Efficient tomography of a quantum many-body system. *Nature Physics*, 13(12):1158–1162, December 2017.

[128] Giacomo Torlai and Roger G. Melko. Machine-Learning Quantum States in the NISQ Era. *Annual Review of Condensed Matter Physics*, 11(1):325–344, March 2020.

[129] C. Kim, J. K. Rhee, W. Lee, and J. Ahn. Mixed Quantum State Dynamics Estimation with Artificial Neural Network. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 740–747, October 2018.

[130] Juan Carrasquilla, Giacomo Torlai, Roger G. Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155, March 2019.

[131] Paul Smolensky. Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 194–281. MIT Press, February 1986.

[132] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

[133] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[134] Xun Gao and Luming Duan. Efficient classical simulation of noisy quantum computation. *arXiv:1810.03176 [quant-ph]*, October 2018.

[135] Kenny Choo, Giuseppe Carleo, Nicolas Regnault, and Titus Neupert. Symmetries and many-body excited states with neural-network quantum states. *Physical Review Letters*, 121(16), October 2018.

[136] Ivan Glasser, Nicola Pancotti, Moritz August, Ivan D. Rodriguez, and J. Ignacio Cirac. Neural-Network Quantum States, String-Bond States, and Chiral Topological States. *Physical Review X*, 8(1):011006, January 2018.

[137] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted Boltzmann machines and tensor network states. *Physical Review B*, 97(8):085104, February 2018.

[138] Yusuke Nomura, Andrew S. Darmawan, Youhei Yamaji, and Masatoshi Imada. Restricted Boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B*, 96(20):205152, November 2017.

[139] Yunqin Zheng, Huan He, Nicolas Regnault, and B. Andrei Bernevig. Restricted Boltzmann machines and matrix product states of one-dimensional translationally invariant stabilizer codes. *Physical Review B*, 99(15):155129, April 2019.

[140] Steven Weinstein. Neural networks as "hidden" variable models for quantum systems. *arXiv:1807.03910 [cond-mat, physics:quant-ph]*, July 2018.

[141] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, April 1982.

[142] H. Chau Nguyen, Riccardo Zecchina, and Johannes Berg. Inverse statistical problems: From the inverse Ising problem to data science. *Advances in Physics*, 66(3):197–261, July 2017. _eprint: https://doi.org/10.1080/00018732.2017.1341604.

[143] R. L. Stratonovich. On a Method of Calculating Quantum Distribution Functions. *Soviet Physics Doklady*, 2:416, July 1957.

[144] L. D. Faddeev and V. N. Popov. Feynman diagrams for the Yang-Mills field. *Physics Letters B*, 25(1):29–30, July 1967.

[145] Francesco Ferrari, Federico Becca, and Juan Carrasquilla. Neural Gutzwiller-projected variational wave functions. *Physical Review B*, 100(12):125131, September 2019.

[146] Faster Parallel Reductions on Kepler, February 2014.

[147] Yoshua Bengio and Olivier Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, June 2009.

[148] Asja Fischer and Christian Igel. Bounding the Bias of Contrastive Divergence Learning. *Neural Computation*, 23(3):664–673, December 2010.

[149] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On Contrastive Divergence Learning. page 8.

[150] Yixuan Qiu, Lingsong Zhang, and Xiao Wang. Unbiased Contrastive Divergence Algorithm for Training Energy-Based Latent Variable Models. In *International Conference on Learning Representations*, September 2019.

[151] Ilya Sutskever and Tijmen Tieleman. On the Convergence Properties of Contrastive Divergence. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 789–795, March 2010.

[152] Alan L. Yuille. The Convergence of Contrastive Divergences. In *NIPS*, 2004.

[153] Matthias Troyer and Uwe-Jens Wiese. Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations. *Physical Review Letters*, 94(17):170201, May 2005.

[154] Matthew B. Hastings, Iván González, Ann B. Kallin, and Roger G. Melko. Measuring Renyi Entanglement Entropy in Quantum Monte Carlo Simulations. *Physical Review Letters*, 104(15):157201, April 2010.

[155] V. Vedral. The Role of Relative Entropy in Quantum Information Theory. *Reviews of Modern Physics*, 74(1):197–234, March 2002.

[156] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition, 2010.

[157] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[158] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, 2018. Association for Computational Linguistics.

[159] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv:1909.11942 [cs]*, February 2020.

[160] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K. Clark, Maksims Volkovs, and Leandro Aolita. Probabilistic Simulation of Quantum Circuits with the Transformer. *arXiv:1912.11052 [cond-mat, physics:quant-ph]*, December 2019.

[161] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. *arXiv:2001.04451 [cs, stat]*, February 2020.

[162] Dan Sehayek, Anna Golubeva, Michael S. Albergo, Bohdan Kulchytskyy, Giacomo Torlai, and Roger G. Melko. The learnability scaling of quantum states: Restricted Boltzmann machines. *arXiv:1908.07532 [quant-ph]*, August 2019.

[163] Tad Hogg and Dmitriy Portnov. Quantum Optimization. *arXiv:quant-ph/0006090*, June 2000.

[164] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. *arXiv:1412.6062 [quant-ph]*, December 2014.

[165] G. C. Wick. Properties of Bethe-Salpeter Wave Functions. *Physical Review*, 96(4):1124–1134, November 1954.

[166] D. C. Handscomb. The Monte Carlo method in quantum statistical mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(4):594–598, October 1962.

[167] R. Blankenbecler and R. L. Sugar. Projector Monte Carlo method. *Physical Review D*, 27(6):1304–1311, March 1983.

[168] Anders W. Sandvik and Juhani Kurkijärvi. Quantum Monte Carlo simulation method for spin systems. *Physical Review B*, 43(7):5950–5961, March 1991.

[169] E. Y. Loh, J. E. Gubernatis, R. T. Scalettar, S. R. White, D. J. Scalapino, and R. L. Sugar. Sign problem in the numerical simulation of many-electron systems. *Physical Review B*, 41(13):9301–9307, May 1990.

[170] Martin C. Gutzwiller. Effect of Correlation on the Ferromagnetism of Transition Metals. *Physical Review Letters*, 10(5):159–162, March 1963.

[171] Martin C. Gutzwiller. Correlation of Electrons in a Narrow s Band. *Physical Review*, 137(6A):A1726–A1735, March 1965.

[172] Hiromi Otsuka. Variational Monte Carlo Studies of the Hubbard Model in One- and Two-Dimensions –Off-Diagonal Intersite Correlation Effects–. *Journal of the Physical Society of Japan*, 61(5):1645–1656, May 1992.

[173] D. Baeriswyl. Variational Schemes for Many-Electron Systems. In Alan R. Bishop, David K. Campbell, Pradeep Kumar, and Steven E. Trullinger, editors, *Nonlinearity in Condensed Matter*, Springer Series in Solid-State Sciences, pages 183–193. Springer Berlin Heidelberg, 1987.

[174] F. Verstraete, D. Porras, and J. I. Cirac. Density Matrix Renormalization Group and Periodic Boundary Conditions: A Quantum Information Perspective. *Physical Review Letters*, 93(22):227205, November 2004.

[175] A. W. Sandvik and G. Vidal. Variational Quantum Monte Carlo Simulations with Tensor-Network States. *Physical Review Letters*, 99(22):220602, November 2007.

[176] G. Vidal. Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension. *Physical Review Letters*, 98(7):070201, February 2007.

[177] Nahuel Freitas, Giovanna Morigi, and Vedran Dunjko. Neural network operations and Susuki–Trotter evolution of neural network states. *International Journal of Quantum Information*, 16(08):1840008, November 2018.

[178] S. Pilati, E. M. Inack, and P. Pieri. Self-learning projective quantum Monte Carlo simulations guided by restricted Boltzmann machines. *arXiv:1907.00907 [cond-mat]*, July 2019.

[179] C. J. Umrigar, K. G. Wilson, and J. W. Wilkins. Optimized trial wave functions for quantum Monte Carlo calculations. *Physical Review Letters*, 60(17):1719–1722, April 1988.

[180] S. Sorella. Green Function Monte Carlo with Stochastic Reconfiguration. *Physical Review Letters*, 80(20):4558–4561, May 1998.

[181] S. Sorella. Generalized Lanczos Algorithm for Variational Quantum Monte Carlo. *Physical Review B*, 64(2), June 2001.

[182] Sandro Sorella. Wave function optimization in the variational Monte Carlo method. *Physical Review B*, 71(24):241103, June 2005.

[183] Tyson Jones, Suguru Endo, Sam McArdle, Xiao Yuan, and Simon C. Benjamin. Variational quantum algorithms for discovering Hamiltonian spectra. *Physical Review A*, 99(6):062304, June 2019.

[184] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014.

[185] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *arXiv:1812.01041 [cond-mat, physics:quant-ph]*, December 2018.

[186] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A*, 97(2):022304, February 2018.

[187] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2):34, February 2019.

[188] Guillaume Verdon, Juan Miguel Arrazola, Kamil Brádler, and Nathan Killoran. A Quantum Approximate Optimization Algorithm for continuous problems. *arXiv:1902.00409 [quant-ph]*, February 2019.

[189] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Physical Review A*, 92(4):042303, October 2015.

[190] Wen Wei Ho and Timothy H. Hsieh. Efficient variational simulation of non-trivial quantum states. *SciPost Physics*, 6(3):029, March 2019.

[191] Wen Wei Ho, Cheryne Jonay, and Timothy H. Hsieh. Ultrafast variational simulation of nontrivial quantum states with long-range interactions. *Physical Review A*, 99(5):052332, May 2019.

[192] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon Benjamin, and Xiao Yuan. Variational quantum simulation of imaginary time evolution. *arXiv:1804.03023 [quant-ph]*, April 2018.

[193] Mario Motta, Chong Sun, Adrian Teck Keng Tan, Matthew J. O' Rourke, Erika Ye, Austin J. Minnich, Fernando G. S. L. Brandao, and Garnet Kin-Lic Chan. Quantum Imaginary Time Evolution, Quantum Lanczos, and Quantum Thermal Averaging. *arXiv:1901.07653 [quant-ph]*, January 2019.

[194] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58(5):5355–5363, November 1998.

[195] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution. January 2000.

[196] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. *Science*, 292(5516):472–475, April 2001.

[197] Boris Altshuler, Hari Krovi, and Jérémie Roland. Anderson localization makes adiabatic quantum optimization fail. *Proceedings of the National Academy of Sciences*, 107(28):12446–12450, July 2010.

[198] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. May 2015.

[199] Zhang Jiang, Eleanor G. Rieffel, and Zhihui Wang. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Physical Review A*, 95(6):062317, June 2017.

[200] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, March 2019.

[201] D. Zhu, S. Johri, N. M. Linke, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Y. Matsuura, T. H. Hsieh, and C. Monroe. Generation of Thermofield Double States and Critical Ground States with a Quantum Computer. *arXiv:1906.02699 [cond-mat, physics:hep-th, physics:quant-ph]*, February 2020.

[202] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Edward Farhi, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Matthew P. Harrigan, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Erik Lucero, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Florian Neukart, Hartmut Neven, Murphy Yuezhen Niu, Thomas E. O'Brien, Bryan O'Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Kevin J. Sung, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, and Leo Zhou. Quantum Approximate Optimization of Non-Planar Graph Problems on a Planar Superconducting Processor. *arXiv:2004.04197 [quant-ph]*, April 2020.

[203] M. B. Hastings. Classical and Quantum Bounded Depth Approximation Algorithms. *arXiv:1905.07047 [quant-ph]*, August 2019.

[204] V. Akshay, H. Philathong, M. E. S. Morales, and J. Biamonte. Reachability Deficits in Quantum Approximate Optimization. *Physical Review Letters*, 124(9):090504, March 2020.

[205] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to State Preparation and Variational Optimization from Symmetry Protection. *arXiv:1910.08980 [cond-mat, physics:quant-ph]*, October 2019.

[206] Takashi Yanagisawa, Soh Koike, and Kunihiko Yamaji. Off-Diagonal Wave Function Monte Carlo Studies of Hubbard Model I. *Journal of the Physical Society of Japan*, 67(11):3867–3874, November 1998.

[207] Takashi Yanagisawa. Crossover from Weakly to Strongly Correlated Regions in the Two-dimensional Hubbard Model — Off-diagonal Wave Function Monte Carlo Studies of Hubbard Model II —. *Journal of the Physical Society of Japan*, 85(11):114707, October 2016.

[208] Takashi Yanagisawa. Antiferromagnetism, Superconductivity and Phase Diagram in the Two-Dimensional Hubbard Model —Off-Diagonal Wave Function Monte Carlo Studies of Hubbard Model III—. *Journal of the Physical Society of Japan*, 88(5):054702, April 2019.

[209] Mohammad-Sadegh Vaezi and Abolhassan Vaezi. A unified theory of variational and quantum Monte Carlo methods and beyond. *arXiv:1810.00864 [cond-mat.str-el]*, October 2018.

[210] S. Bravyi, M. B. Hastings, and F. Verstraete. Lieb-Robinson Bounds and the Generation of Correlations and Topological Quantum Order. *Physical Review Letters*, 97(5):050401, July 2006.

[211] Elliott Lieb, Theodore Schultz, and Daniel Mattis. Two soluble models of an antiferromagnetic chain. *Annals of Physics*, 16(3):407–466, December 1961.

[212] Subir Sachdev. *Quantum Phase Transitions*. Cambridge university press, second edition, 2011.

[213] Ieong Wong, Wenjia Liu, Chih-Ming Ho, and Xianting Ding. Continuous Adaptive Population Reduction (CAPR) for Differential Evolution Optimization. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 22(3):289–305, June 2017.

[214] Lu Liu, Anders W. Sandvik, and Wenan Guo. Typicality at quantum-critical points. *Chinese Physics B*, 27(8):087501, August 2018.

[215] Ingo Peschel. Calculation of reduced density matrices from correlation functions. *Journal of Physics A: Mathematical and General*, 36(14):L205–L208, April 2003.

[216] G. Vidal, J. I. Latorre, E. Rico, and A. Kitaev. Entanglement in quantum critical phenomena. *Physical Review Letters*, 90(22), June 2003.

[217] J. I. Latorre, E. Rico, and G. Vidal. Ground State Entanglement in Quantum Spin Chains. *Quantum Info. Comput.*, 4(1):48–92, January 2004.

[218] Pasquale Calabrese and John Cardy. Evolution of entanglement entropy in one-dimensional systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(04):P04010, April 2005.

[219] Michaël Mariën, Koenraad M. R. Audenaert, Karel Van Acoleyen, and Frank Verstraete. Entanglement Rates and the Stability of the Area Law for the Entanglement Entropy. *arXiv:1411.0680 [math-ph]*, November 2014.

[220] Pasquale Calabrese and John Cardy. Entanglement and correlation functions following a local quench: A conformal field theory approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(10):P10004–P10004, October 2007.

[221] Anders W. Sandvik. Ground State Projection of Quantum Spin Systems in the Valence-Bond Basis. *Physical Review Letters*, 95(20):207203, November 2005.

[222] Anders W. Sandvik. Stochastic series expansion method for quantum Ising models with arbitrary interactions. *Physical Review E*, 68(5):056701, November 2003.

[223] Roger G. Melko. Stochastic Series Expansion Quantum Monte Carlo. In Adolfo Avella and Ferdinando Mancini, editors, *Strongly Correlated Systems*, volume 176, pages 185–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[224] Tatiana Tommasi, Novi Patricia, and Tinne Tuytelaars. A Deeper Look at Dataset Bias. page 19.

[225] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91, January 2018.

[226] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of Opportunity in Supervised Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc., 2016.

[227] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning Not to Learn: Training Deep Neural Networks with Biased Data. *arXiv:1812.10352 [cs]*, April 2019.

[228] Sebastian J. Wetzel and Manuel Scherzer. Machine learning of explicit order parameters: From the Ising model to SU(2) lattice gauge theory. *Physical Review B*, 96(18):184410, November 2017.

[229] Sebastian J. Wetzel, Roger G. Melko, Joseph Scott, Maysum Panju, and Vijay Ganesh. Discovering Symmetry Invariants and Conserved Quantities by Interpreting Siamese Neural Networks. *arXiv:2003.04299 [cond-mat, physics:physics]*, March 2020.

[230] Anna Dawid, Patrick Huembeli, Michał Tomza, Maciej Lewenstein, and Alexandre Dauphin. Phase Detection with Neural Networks: Interpreting the Black Box. *arXiv:2004.04711 [cond-mat, physics:quant-ph]*, April 2020.

[231] Yi Zhang, Paul Ginsparg, and Eun-Ah Kim. Interpreting machine learning of topological quantum phase transitions. *Physical Review Research*, 2(2):023283, June 2020.

[232] Joaquin F. Rodriguez-Nieva and Mathias S. Scheurer. Identifying topological order through unsupervised machine learning. *Nature Physics*, 15(8):790–795, August 2019.

[233] Haribabu Kandi, Ayushi Jain, Swetha Velluva Chathoth, Deepak Mishra, and Gorthi R. K. Sai Subrahmanyam. Incorporating rotational invariance in convolutional neural network architecture. *Pattern Analysis and Applications*, 22(3):935–948, August 2019.

[234] Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. *arXiv:1902.04615 [cs, stat]*, May 2019.

[235] Taco S. Cohen and Max Welling. Group Equivariant Convolutional Networks. *arXiv:1602.07576 [cs, stat]*, June 2016.

[236] Zhiyuan Zhang, Binh-Son Hua, David W. Rosen, and Sai-Kit Yeung. Rotation Invariant Convolutions for 3D Point Clouds Deep Learning. *arXiv:1908.06297 [cs]*, August 2019.

[237] Bálint Joó, Chulwoo Jung, Norman H. Christ, William Detmold, Robert G. Edwards, Martin Savage, and Phiala Shanahan. Status and Future Perspectives for Lattice Gauge Theory Calculations to the Exascale and Beyond. *The European Physical Journal A*, 55(11):199, November 2019.

[238] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), September 2019.

[239] M. S. Albergo, G. Kanwar, and P. E. Shanahan. Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Physical Review D*, 100(3):034515, August 2019.

[240] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Recurrent Neural Network Wavefunctions. *arXiv:2002.02973 [cond-mat, physics:physics, physics:quant-ph]*, February 2020.

[241] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, April 2015.

[242] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968 [physics, physics:quant-ph]*, February 2020.

[243] Andrea Mari, Thomas R. Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran. Transfer learning in hybrid classical-quantum neural networks. *arXiv:1912.08278 [quant-ph, stat]*, December 2019.

[244] Guillaume Verdon, Jacob Marks, Sasha Nanda, Stefan Leichenauer, and Jack Hidary. Quantum Hamiltonian-Based Models and the Variational Quantum Thermalizer Algorithm. *arXiv:1910.02071 [quant-ph]*, October 2019.

[245] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyeon Yoo, Sergei V. Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, Martin Leib, Andrea Skolik, Michael Streif, David Von Dollen,

Jarrod R. McClean, Sergio Boixo, Dave Bacon, Alan K. Ho, Hartmut Neven, and Masoud Mohseni. TensorFlow Quantum: A Software Framework for Quantum Machine Learning. *arXiv:2003.02989 [cond-mat, physics:quant-ph]*, March 2020.

APPENDICES

Appendix A

Quantum Ising model sampler

The one-dimensional transverse-field Ising model (TFIM) is prototypical example of an exactly solvable interacting quantum system [211]. It is described by nearest neighbour interactions between spin projections along the z axis and a magnetic field h along the x-axis projection:

$$\hat{H} = -J \sum_{i} \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z - h \sum_{i} \hat{\sigma}_i^x \, , \tag{A.1}$$

where $\hat{\sigma}_i^{x/z}$ are spin-1/2 Pauli operators acting on site i, and we assume periodic boundary conditions.

This model has $\mathbb{Z}_2$ symmetry apparent by replacing $\hat{\sigma}_i^a \to -\hat{\sigma}_i^a$ for $a = \{x, y, z\}$. The transverse $hat\sigma_i^x$ term introduces quantum fluctuations. The system undergoes a quantum phase transition at critical field strength h_c. Below h_c, the ground-state is degenerate with two solutions: all $\hat{\sigma}_i^z$ spins are aligned either up or down. The local order-parameter $M = \langle \hat{\sigma}_i^z \rangle$ is non-zero in this region. Conversely, for $h > h_c$, the system obeys the Z_2 symmetry so that $M = 0$. This paramagnetic (PM) phases is called disordered.

The TFI model in one-dimension admits a dual representation as non-interacting spinless fermions [211]. The most general form of a non-interacting fermion Hamiltonian is:

$$H = 2 \sum_{i,j} c_i^\dagger A_{i,j} c_j + \frac{1}{2} c_i^\dagger B_{i,j} c_j^\dagger + \text{h.c.} \tag{A.2}$$

For the specific case of the Ising model, it works out to be

$$A_{i,i} = -1, \tag{A.3}$$

$$A_{i+1,i} = A_{i,i+1} = -\frac{1}{2h}, \tag{A.4}$$

$$B_{i,i+1} = -B_{i+1,i} = -\frac{1}{2h} \tag{A.5}$$

As a free system, all correlation functions can be determined from Wick's theorem by using only the correlation, $\langle c^\dagger c \rangle$, and anomalous correlation functions, $\langle c^\dagger c^\dagger \rangle$ [165, 215]. A corollary is the ground state is completely characterized by the covariance matrix

$$F_{i,j} = \langle c_i^\dagger c_j^\dagger \rangle + \langle c_i^\dagger c_j \rangle - \langle c_i c_j \rangle - \langle c_i c_j^\dagger \rangle \ . \tag{A.6}$$

The wavefunction is then $|\psi\rangle = \exp\left(\sum_{i,j} F_{i,j} c_i^\dagger c_j^\dagger\right) |SD\rangle$ where $|SD\rangle$ is the anti-symmetrized Slater determinant state [12].

For the exact ground state, the covariance matrix simplifies to

$$F_{i,j} = \frac{2}{L} \sum_{k=0}^{(L-1)/2} \frac{u_k}{v_k} \sin(|i-j|\theta_k) \tag{A.7}$$

where $\theta_k = (2k+1)\pi/L$ and u_k, v_k are

$$u_k = \sqrt{\frac{E_k + \zeta_k}{2E_k}}, \qquad v_k = i\sqrt{\frac{E_k - \zeta_k}{2E_k}}, \tag{A.8}$$

$$E_k = \sqrt{J^2 + h^2 + 2Jh\cos\theta_k}, \tag{A.9}$$

$$\zeta_k = -h - J\cos k \, . \tag{A.10}$$

This makes it possible to samples configurations of spins effectively using the Metropolis-Hastings algorithm [12]. A spin configuration $\tilde{\mathbf{x}}$ is proposed from $\mathbf{x}$ by flipping one or more spins randomly. The proposed $\tilde{\mathbf{x}}$ is accepted if a random number r, drawn from the range $[0, 1]$ is less than the min $\left(1, \frac{\det F_{\tilde{\mathbf{x}},\tilde{\mathbf{x}}}}{\det F_{\mathbf{x},\mathbf{x}}}\right)$ where $F_{\mathbf{x},\mathbf{x}}$ is the covariance matrix restricted to the rows and columns of $\boldsymbol{x}$ which are non-zero. The fast update trick can be used to reduce the updates cost from $O(N^3)$ time to $O(N)$ [12]. The probability of a sample is given by the Born rule: $p(\mathbf{x}) = |\langle \mathbf{x}|\psi\rangle|^2 = \det(F_{\mathbf{x},\mathbf{x}})/Z^2$ where $Z = \det(I + F)$ is the partition function. Notice that in the σ^z basis, the wavefunction $\psi(\boldsymbol{x})$ contains only non-negative elements.