

# DEEP PHYSICS

CHAITANYA KUMAR

*B.Tech, Computer Science Engineering*

A THESIS SUBMITTED FOR THE DEGREE OF MASTERS IN COMPUTING  
(M.COMP.), SPECIALISATION IN ARTIFICIAL INTELLIGENCE


SCHOOL OF COMPUTING  
NATIONAL UNIVERSITY OF SINGAPORE

2021

## Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

A handwritten signature in black ink, appearing to be 'Chaitanya Kumar', written above a horizontal line.

Chaitanya Kumar

November 2021

## Acknowledgments

I would like to express my sincere gratitude to Prof. Stephane Bressan, for his guidance and encouragement all along. His advice and constructive criticism were key in getting this work to its present state.

I am equally thankful for the patience and support provided by Remmy, Stephane's doctoral student and my collaborator. Being new to the problem space and lacking the background, I had several queries and he helped getting them addressed.

I am also grateful to Prof. Dario Poletti (Associate Professor, SUTD) for his insights as a physicist. Those were instrumental in helping me realise the motivations backing our research decisions.

Lastly, I extend my heartfelt thanks to my friends and family for bearing with me when I was in the thick of it and supporting me when I wasn't so sure of myself.

## Abstract

Recent advances in Machine Learning have made it a go-to suite of techniques for various disciplines. Quantum many-body physics which looks at systems with many fundamental particles at the scale of atoms and below, is one such discipline. The information about the dynamics of such systems is stored in the wave function. However, the number of states of such a system grows exponentially in the number of particles in the system. This makes the study of such systems extremely hard, as they require a large amount of memory to store the entire wavefunction.

Carleo and Troyer, in their recent work, proposed a framework, neural network quantum states, to solve quantum many-body physics problems using neural networks trained in an unsupervised manner.

In this thesis, we study the energy of quantum many-body models in two challenging scenarios, the ground state of the Ising  $J_1 - J_2$  model and low-lying excited states for the Ising model.

Firstly, we evaluate the existing  $(k, p)$ -tiling transfer learning protocols for increasing the scalability, efficiency and effectiveness of neural network quantum states for the Ising  $J_1 - J_2$  model. We test the protocols for restricted Boltzmann machine and extend the evaluation to multilayer perceptron as well. Transfer learning reuses a machine learning model trained for a specific task to initialise another model that is subsequently tuned for a similar but different task. We are able to show that these protocols improve the scalability, efficiency and effectiveness of neural network quantum states, when determining the ground state.

Secondly, we explore and evaluate two approaches to determine excited states for the Ising model. To that end, we employ different techniques to rearrange the eigenvalues of the Hamiltonian matrix such that the lowest eigenvalue corresponds to the desired excited state energy level. We evaluate both the approaches in terms of their effectiveness as well as their efficiency. Our initial experiments with these show that despite being fundamentally similar approaches (in that they modify the Hamiltonian matrix), they differ in their efficiency as well as scalability.

# Contents

<b>List of Figures</b>	vii
<b>List of Tables</b>	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions	1
1.2 Outline of the thesis	2
<b>2 Background and Related Work</b>	<b>3</b>
2.1 Quantum Many Body Physics	3
2.1.1 Quantum Many Body Problem	4
2.1.2 Quantum Many Body Models	7
2.2 Machine Learning and Neural Networks	9
2.2.1 Restricted Boltzmann Machine	9
2.2.2 Multilayer Perceptron	11
2.2.3 Transfer Learning	13
2.3 Neural-network Quantum States (NQS)	13
2.3.1 Restricted Boltzmann Machine (RBM) for NQS	15
2.3.2 Multilayer Perceptron (MLP) for NQS	16
2.4 The NQS library	17
2.5 Related Work	18
<b>3 Ground State Energy for Ising <math>J_1 - J_2</math> model using NQS</b>	<b>21</b>
3.1 Transfer Learning for finding Ground State Energy	21
3.1.1 Transfer Learning protocols for scalability	22
3.2 Performance Evaluation	24
3.2.1 Ground State Energy using RBM	26
3.2.2 Ground State Energy using MLP	28
3.3 Conclusions	30
<b>4 Excited States Energy levels for Ising model using NQS</b>	<b>37</b>
4.1 Excited States Methods	37
4.1.1 Excited States using Method-I	38
4.1.2 Excited States using Method-II	43
4.2 Performance Evaluation	45
4.2.1 Efficiency	46
4.2.2 Effectiveness	47

---

4.3	Conclusions	48
<b>5</b>	<b>Conclusion and Future Work</b>	<b>49</b>
5.1	Conclusion	49
5.2	Future Work	50
<b>6</b>	<b>Appendix</b>	<b>62</b>
6.1	Example Hamiltonian Matrix	62
6.2	Expression for $E_{loc}(x)$	66
6.3	Transfer Learning across parameters for Ising $J_1 - J_2$ model	67
6.3.1	Transfer Learning across parameters using RBM	68
6.3.2	Transfer Learning across parameters using MLP	68

## List of Figures

- 2.1 Example of a one-dimensional many-body system with 9 particles and the  $J_1$  and  $J_2$  interactions between them 8
- 2.2 The architecture of a restricted Boltzmann machine with  $L_v$  visible nodes and  $L_h$  hidden nodes. The solid lines denote the weights  $\mathbf{W}$ , the dotted lines denote the visible biases  $\mathbf{a}$  and the dashed lines denote the hidden biases  $\mathbf{b}$ . 10
- 2.3 The architecture of a multilayer perceptron with  $L_v$  visible nodes and one hidden layer. The visible layer consists of visible nodes  $x_1, \dots, x_{L_v}$ . The connections between any two consecutive layers are given by the weight matrix  $W$ . The solid lines denote the weights  $\mathbf{W}$ , while the dashed lines denote the biases  $\mathbf{b}$ . 11
- 3.1 Schematic representation of the different transfer learning protocols used to scale up one-dimensional systems.  $N, N_h$  and  $N', N'_h$  correspond to the number of visible and hidden nodes for the base and target networks, respectively. The different panels show how to construct the target weight matrix  $\mathbf{W}$  from the base weight matrix  $\tilde{\mathbf{W}}$  by replicating the colored rows and filling the grey cells with randoms entries. Panels (a), (b) and (c):  $(1, 2) - \text{tiling}$ ,  $(2, 2) - \text{tiling}$  and  $(L, 2) - \text{tiling}$ , respectively. Here the scaling factor is 2. Panel (d) shows the  $(L, 4) - \text{tiling}$  used when scaling up the network by a factor 4. Image and caption from [102]. 24
- 3.2 Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = -0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 27
- 3.3 Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = -0.5, J_2 = 5.0, h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 28

- 3.4 Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = 0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 29
- 3.5 Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = 0.5, J_2 = 5.0, h = 1.0$ , i.e., *ferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 29
- 3.6 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 30
- 3.7 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = 5.0, h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 31
- 3.8 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 32
- 3.9 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5, J_2 = 5.0, h = 1.0$ , i.e., *ferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 32



- 3.10 The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = -0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 33
- 3.11 The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = -0.5, J_2 = 5.0, h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 33
- 3.12 The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = 0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 34
- 3.13 The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = 0.5, J_2 = 5.0, h = 1.0$ , i.e., *ferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 34
- 3.14 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 35
- 3.15 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = 5.0, h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 35

- 3.16 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 36
- 3.17 Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5, J_2 = 5.0, h = 1.0$ , i.e., *ferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. 36
- 4.1 Time to convergence for Method-I (shown in red) and Method-II (shown in green) across the different  $J/h$  values, with 4 and 8 spins, respectively. The error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 10 realisations. 46
- 4.2 Relative first excited state energy error  $\Delta E/E_1$  for different values of  $J/h$  using Method-I (shown in red) and Method-II (shown in green), with 4 and 8 spins, respectively. The error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 10 realisations. 47
- 4.3 Tables listing the Mean Overlap values for the overlap between the ground state, and the first excited state obtained using Method-I, averaged over 10 iterations, for systems with 4 and 8 spins, respectively 48
- 4.4 Tables listing the Mean Overlap values for the overlap between the ground state, and the first excited state obtained using Method-II, averaged over 10 iterations, for systems with 4 and 8 spins, respectively 48
- 5.1 Exact wave function plots for  $N = 4, J/h = 0.5$ . Note the asymmetry in the excited state wave functions as opposed to the symmetry in the ground state wave function 51
- 6.1 Effectiveness (left column) and Efficiency (right column) among cold-start and transfer learning runs, for systems with  $N = 8$  and  $N = 16$ , with  $h = 1.0, J_1 = -0.5$  and  $J_2 = \{-1.0, -2.0, -3.0\}$  for Restricted Boltzmann Machine. The red bars and lines indicate cold-start, while the green ones indicate transfer learning runs. The error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, across 20 iterations. 69

- 6.2 Effectiveness (left column) and Efficiency (right column) among cold-start and transfer learning runs, for systems with  $N = 8$  and  $N = 16$ , with  $h = 1.0$ ,  $J_1 = -0.5$  and  $J_2 = \{-1.0, -2.0, -3.0\}$  for Multilayer Perceptron. The red bars and lines indicate cold-start, while the green ones indicate transfer learning runs. The error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, across 20 iterations.

## List of Tables

# CHAPTER 1

## Introduction

Condensed matter physics is concerned with the changes that materials undergo on account of external physical properties such as temperature and magnetism. These changes can typically be modeled through particle-level interactions. There exist various mathematical models and algorithms to capture these interactions. However, these algorithms have complexity exponential in terms of the input system size. As a result, while the conventional techniques have notable results, they have limitations.

The ability of modern machine learning to identify, classify and analyse huge datasets of images and text, makes it suitable to this paradigm. In fact, machine learning has already been employed as a tool by condensed matter physicists for various problems [5, 49, 44, 34, 82, 59]. Specific to the context of this thesis, Carleo and Troyer employed Artificial Neural Networks (ANNs) to capture the intricacies of the quantum many-body system and found the use of ANNs a viable approach, not just for ground-state properties [12] but also for modeling the evolution of the system, by a complex set of excited quantum states [17].

In the context of ground states, Zen et al. propose in [101] and evaluate in [102] transfer learning in both supervised and unsupervised settings, onto Neural-network Quantum States (NQS), using an analytically constructed restricted Boltzmann machine (RBM, see 2.2.1). This thesis leverages the protocols put forth by Zen et. al in [102] and extends them to the Ising  $J_1 - J_2$  model as well as the Multilayer Perceptron (MLP, see 2.2.2) architecture. Additionally, this work also explores using NQS to determine low-lying excited states for the simple Ising model.

### 1.1 Contributions

The key contributions of the work in this thesis are as follows:

- **Ground State in the Ising  $J_1 - J_2$  model using NQS:** [102] establishes the viability of transfer learning in NQS for determining ground state of simple Ising models, using the NQS-IT and NQS-ITT constructions detailed in [101]. This thesis employs said constructions to the Ising  $J_1 - J_2$  model, for both MLP and RBM architectures. The Ising  $J_1 - J_2$  model (explained in Section 2.1.2) takes into account not just the nearest

neighbor, but also the next-nearest neighbor particle interactions. The details for this thread of work are presented in Chapter 3.

- **Excited States for the simple Ising model using NQS:** Given the relevance of the ground state of a system, determining the excited states closer to the ground state energy becomes a natural follow-up thread of inquiry. In that vein, this work employs NQS to find excited states in the one-dimensional Ising model. The approaches to do so start with the ground state and iteratively determine subsequent excited states. Chapter 4 goes into the details of this approach as well as our findings.

## 1.2 Outline of the thesis

The rest of the dissertation is organized as follows:

- Chapter 2 goes into the Quantum Mechanics and Machine Learning background relevant to this work. Specifically, Sec. 2.1 describes the quantum many-body problem, the conventional methods that are used to solve it, and the specific many-body models that we consider in this work. Next, Sec. 2.2 describes the restricted Boltzmann machine (RBM) and multilayer perceptron (MLP) architectures, as well as transfer learning in general. This is then followed by adding the neural quantum states context to these RBM and MLP architectures in Sec. 2.3. Lastly, Sec. 2.5 describes various works from the literature relevant to this thesis.
- Chapter 3 elaborates upon the first contribution of this thesis, i.e., determining ground state for the Ising  $J_1 - J_2$  model. Sec. 3.2 details the evaluation setup as well as findings and Sec. 3.3 summarises them.
- Chapter 4 goes into the two methods proposed for the second contribution, i.e., finding low-lying excited states for the Ising model. Sec. 4.2 evaluates the two methods and reports our findings, summarising in Sec. 4.3.
- Finally, Chapter 5 summarises the work in this thesis, concluding the findings in Sec. 5.1 and enumerating potential future work threads in Sec. 5.2.

# CHAPTER 2

## Background and Related Work

This chapter provides the necessary background for understanding the scope of, as well as contributions made, by this work. In the first section concepts from quantum physics, relevant to the thesis are discussed. This is followed by an introductory description of relevant neural network architectures and techniques in Sec. 2.2. Both of these pave way for understanding Neural-network Quantum States (NQS) [12], which sets the context for this work. Finally, Sec. 2.5 describes the body of literature that motivates the threads of inquiry pursued in this thesis.

### 2.1 Quantum Many Body Physics

In contrast to classical mechanics where a system exists in only one state, a quantum mechanical system may exist as a superposition of multiple states. This information is encoded in what is known as the *wave function* of the system, denoted by  $|\Psi\rangle$ . The set of all possible wave functions for a system forms a vector space, called *Hilbert space*, which is complex and separable. A quantum system consists of interacting particles evolving in a discrete or continuous  $D$ -dimensional space. In general, a particle has two types of characteristics, external and internal. The external characteristics include position and momentum, while the internal ones include magnetic moment, which is proportional to its angular momentum and is also known as spin.

In this work, we focus on particle spin and consider identical particles pinned at the nodes of a  $D$ -dimensional lattice, considering interactions between the nearest and next-nearest neighboring particles. If  $n_s$  is the number of possible spin states, then for a system with  $N$  particles, the size of the Hilbert Space,  $n_s^N$  is exponential in the number of particles. Specifying the value of the spin for each particle in the system provides a *configuration* of the system. The total possible number of configurations is the same as the size of the Hilbert Space, i.e.,  $n_s^N$ . For the context of this thesis, we only consider time-independent wave functions with a discrete and finite (preferably orthonormal) basis, the configuration basis. The wave function is therefore a linear combination of all the basis vectors, denoted by  $|x\rangle$ , associated with each possible configuration  $x$  as shown in Eq. 2.1. Here  $\Psi(x_i)$  is the

projection of  $|\Psi\rangle$  along the basis  $|x_i\rangle$ , and is also known as the amplitude.

$$|\Psi\rangle = \sum_i \Psi(x_i) |x_i\rangle \quad (2.1)$$

The modulus square of a wave function  $|\Psi(x_i)|^2$  is interpreted as the probability density or mass of finding the system in configuration  $x_i$ . This implies that  $\sum_i |\Psi(x_i)|^2 = 1$  for finite discrete  $x_i$ , which is the setup we consider in this work.

A system can either have periodic or open boundary conditions. The particles at the border of a system with periodic boundary conditions are connected. A system with this condition can be thought of as a circle and a torus for one and two dimensional systems, respectively. This boundary condition approximates infinite systems better. On the other hand, a system with open boundary conditions is one where the particles at the border are not connected. We restrict ourselves to quantum many-body<sup>1</sup> systems, which refers to quantum systems with large  $N$ , and open boundary conditions.

The *observables*, for example energy and momentum, are represented as a Hermitian operator acting on the wave function. As the observables are used to study properties of the wave function of a system, we can treat the wave function as a column vector, and the operator as a matrix mapping the wavefunction to other wavefunctions in the Hilbert space. The eigenvalues of the operators are the physical values that the observable can take, as a result of the measurement. Mathematically, when Observable  $O$  operates on a system, the corresponding eigenvalues,  $o_i$ , are the possible results, such that  $O |\Psi_i\rangle = o_i |\Psi_i\rangle$ .

The *Hamiltonian* of a system fully describes its dynamics, the number of particles, interactions between them, and external fields. The Hamiltonian is a Hermitian matrix of size  $n_s^N \times n_s^N$ . The element  $H_{i,j}$  is the projection  $H \mathbf{x}_j$  along basis vector  $\mathbf{x}_i$  meaning that  $H_{i,j} = \mathbf{x}_i^\dagger H \mathbf{x}_j$  or equivalently,  $\langle \mathbf{x}_i | H | \mathbf{x}_j \rangle$ . Further, the eigenvalues of the Hamiltonian are the possible energies of the system while the eigenvectors are the only possible states in which the system can be found after measuring its energy. More details about the Hamiltonian operators we consider are provided in Sec. 2.1.2.

### 2.1.1 Quantum Many Body Problem

$$H |\Psi_i\rangle = E_i |\Psi_i\rangle \quad (2.2)$$

Schrödinger's equation is given by Eq. 2.2, where  $H$  is the Hamiltonian operator,  $E_i$  are the possible energy values of the system where  $E_0 \leq E_1 \leq \dots$  and  $|\Psi_i\rangle$  are the wave functions corresponding to those  $E_i$ . Finding a solution for  $i = 0, 1, \dots$ , solves the many body problem to determine the various energy levels a quantum system can exist in as

---

<sup>1</sup>Here, *many* is usually defined as systems with more than two particles. There also exist quantum few-body systems, which have less than ten particles that lie on a continuous space commonly studied in nuclear physics.



well as the eigenvectors using which one can figure out all the other properties of a system. The Hamiltonian in the equation depends on the system being observed and since it is a Hermitian matrix, its eigenvalues or energy  $E_i$  are real numbers, and the eigenvectors  $|\Psi_i\rangle$  are orthogonal to each other. (Note that two or more eigenvectors may have the same eigenvalue, implying degenerate states.)

Usually, we are interested in the *ground state*,  $|\Psi_0\rangle$  of the quantum system, which is the state with the lowest energy  $E_0$ , the ground state energy. At the ground state, the effects of quantum mechanics are more pronounced and easier to observe, as compared to excited states [52]. Mathematically, we aim to find the eigenstate corresponding to the lowest eigenvalue that solves Eq. 2.2. Non-ground states are termed as *excited states*. Low-lying excited states are of interest to study *quantum dynamics*, i.e., how the system evolves over time. Note that quantum dynamics is beyond the scope of this work, and we only consider time-independent wave functions.

When solving the quantum many-body problem as described by Eq. 2.2,  $H$  is a  $n_s^N \times n_s^N$  square matrix which grows exponentially in system size –  $H_{16 \times 16}, H_{32 \times 32}, H_{64 \times 64}$  for spin-half particle systems of sizes 4, 5 and 6, respectively, for example. While storing the Hamiltonian is often not required (and for several models, including the ones studied in this thesis, it is a sparse matrix), the exponential growth still makes computations very expensive. We also need to compute the eigenstates, which are vectors of size  $n_s^N$ , again exponential in the system size. Given the expensive computation costs, it is challenging to encode the many-body wave function into a computationally tractable form. It becomes imperative then to rely on approximation methods.

The **Exact Diagonalisation** method [29] solves the eigenvalue problem in Eq. 2.2, i.e., finds  $E$  and  $|\Psi\rangle$  that solves Eq. 2.3.

$$(H - E_i \mathbf{I}) |\Psi_i\rangle = 0 \quad (2.3)$$

The solution to Eq 2.3 consists of  $n_s^N$  eigenvalues and eigenvectors and this method becomes infeasible for large matrices. As a reference, for a system of size 32, there are about  $4.29 \times 10^9$  eigenvectors which needs several zetabytes ( $10^{21}$ ) of memory, assuming we use double precision floating point numbers. This method is available for sparse matrices in the **SciPy Python** library.

**Matrix Product State**(MPS) is a scheme to represent the probability amplitudes of a wave function by decomposing them into a product of matrices with a number of indices depending on the geometry in the lattice of the system being observed. For example, Eq. 2.4

shows a state of a one-dimensional system represented by matrix product state.

$$|\Psi\rangle_{MPS} = \sum_i \left( \sum_{a_1, \dots, a_N} A_{a_1, a_2}^{x_1} A_{a_2, a_3}^{x_2} \dots A_{a_{N-1}, a_N}^{x_N} \right) |\mathbf{x}_i\rangle \quad (2.4)$$

where  $A^{x_j}$  is a  $B \times B$  matrix with indices  $a_{j-1}, a_j$  and  $B$  is called the bond dimension.

Every state can be represented by a matrix product state when  $B$  is sufficiently large and becomes exact as  $B \rightarrow \infty$ . It has been shown theoretically and analytically that it is possible to keep  $B$  bounded at a maximum bond dimension value and still represent a large set of physical states very accurately, if not exactly [28, 74]. There is no sampling needed in this method and the goal is to find the matrix  $A^{x_j}$  that approximates the target state. The maximum bond dimension,  $B_B$ , is taken in as a parameter of MPS methods. In terms of memory, MPS requires  $N n_s^2 B_B^2$ , which is linear in the size of the system. MPS methods are readily available in several libraries including ITensor [31] which uses C++ and Julia, Uni10 [45] and ALPS [7] which use C++, and TensorNetwork [75] which uses Python with GPU support.

Note that we use the Exact Diagonalisation and MPS methods described above to get reference energy values for comparison against energy values obtained analytically using the methods proposed in this work.

**Quantum Monte Carlo** (QMC) techniques use the Monte Carlo methods to handle the intractable summation (or integration) in the formulation of many body problems. This is typically done by transforming the quantum mechanical expectation value into a form that can be evaluated stochastically with the Monte Carlo method—by sampling a finite number of configurations. Owing to its stochastic nature, the error of the expectation scales with the number of samples  $N_s$  as  $1/\sqrt{N_s}$ . The QMC method is commonly used to approximate the properties of quantum many-body systems by computing the value of the energy which is the expectation of the Hamiltonian (or the expected value of other observables). In this thesis we use *variational Monte Carlo*, which is one of the many QMC methods. It is commonly used to study the ground state and excited states of the system at zero temperature. The variational Monte Carlo method relies on the variational principle from quantum mechanics and Monte Carlo sampling. The variational principle states that the expected energy of a given Hamiltonian computed from any normalised wave function is always greater than or equal to the ground state. This implies that the problem of finding the ground state can be posed as an optimisation problem that starts with a parameterised trial wave function and then minimising the expected value of the energy to update the parameter. The downside of quantum Monte Carlo methods is that they may need a lot of time to converge and convergence relies on the number of samples. QMC methods are available in several libraries, for instance, the ALPS library [7], the CASINO library [64] and the QMCPACK library [47]. Neural Network quantum states [12] belong to the family of variational Monte Carlo methods that use neural networks as the trial wave function.

### 2.1.2 Quantum Many Body Models

In this section, we describe the two quantum physics models relevant for the work presented in this thesis. Sec. 2.1.2 details the relatively simpler, Ising model, which we consider when determining low-lying excited states. The Ising  $J_1 - J_2$  model is described in Sec. 2.1.2 and is the one we evaluate when determining the ground state.

#### Transverse Field Ising model

In this mathematical model, all particles form a  $d$ -dimensional lattice (graph) and carry a binary discrete spin. Each spin is in one of two states: up (represented by  $+1$ ) or down (represented by  $-1$ ). As each particle has two states, the number of configurations for  $N$  particles equals  $2^N$ . A configuration  $x$  is given by the value of spin at each site, i.e.,  $x = (x_1, x_2, \dots, x_N)$  where  $x_i = \pm 1$ . Throughout the rest of the thesis, we refer to this model as the Ising model.

Additionally,

- each spin interacts with its nearest neighbors along the  $z$ -axis
- there exists an external magnetic field applied along the  $x$ -axis, interacting with all spin

The Hamiltonian for this model is given by Eq. 2.5

$$H_I = -h \sum_i^n \sigma_i^x - J \sum_i^n \sum_{j \in \text{neigh}(i)} \sigma_i^z \sigma_j^z \quad (2.5)$$

where  $\sigma_i^\alpha$  and  $\sigma_j^\alpha$  are the Pauli matrices where  $\alpha = \{x, y, z\}$ ;  $i$  and  $j$  indicates the position of the spin it is acting upon, and  $\text{neigh}(\cdot)$  returns the nearest neighboring sites. An example of how the Hamiltonian matrix is populated is given in Sec. 6.1 in the Appendix.

Only the relative strength between  $J$  and  $h$  is of significance. For example, an Ising model with  $h = 1$  and  $J = 1$  has the same static properties as another with  $h = 2$  and  $J = 2$ , except that the latter has twice the energy. Therefore, we can regard  $J/|h|$  as the parameter of the system.

**Note:** In general,  $J$  can have different values for each pair of particles but it is a common simplification to assume that all of the nearest neighbors have the same interaction strength. Similarly,  $h$  can, theoretically, have different values for each particle, but as an external magnetic field is typically of the same magnitude everywhere, we use  $h$  instead of  $h_{\langle i,j \rangle}$ .

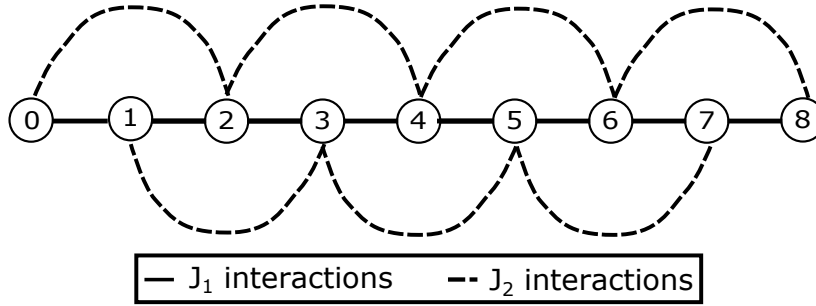
#### Ising $J_1 - J_2$ model

In addition to the Ising model described above, some experiments were also run on the Ising  $J_1 - J_2$  model. The latter factors in the next-nearest neighbor interactions in addition to

the nearest neighbor ones (See Figure 2.1). There are three parameters in this model:  $h$  representing the interaction with the magnetic field,  $J_1$  representing the nearest neighbor interactions and  $J_2$  representing the next-nearest neighbor interactions. At  $J_2 = 0$  the model reduces to the Ising model. The Hamiltonian for the Ising  $J_1 - J_2$  model is given by Eq. 2.6.

$$H = -h \sum_i \sigma_i^x - J_1 \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - J_2 \sum_{\langle\langle i,j \rangle\rangle} \sigma_i^z \sigma_j^z \quad (2.6)$$

where  $\langle i,j \rangle$  denotes that  $i$  and  $j$  are nearest neighbours,  $\langle\langle i,j \rangle\rangle$  denotes that  $i$  and  $j$  are next-nearest neighbours, and  $\sigma_i^x$  and  $\sigma_j^z$  are the Pauli operators acting on site  $i$  and  $j$  respectively.



**Figure 2.1:** Example of a one-dimensional many-body system with 9 particles and the  $J_1$  and  $J_2$  interactions between them

As part of this thesis, we evaluated our system to find the ground state energy for different values of the parameters—  $h = 1$ ,  $J_1 = \{-0.5, 0.5\}$  and  $J_2 = \{-5.0, 5.0\}$ , with open boundary conditions. These correspond to the different frustrated and non-frustrated phases for the Ising  $J_1 - J_2$  model:

1. when  $J_1 < 0$  and  $J_2 \gg J_1/2$  and both  $J_1$  and  $J_2$  dominate  $h$ , the nearest neighbors want to be aligned anti-parallel to each other due to  $J_1$  and the next-nearest neighbors want to be aligned parallel to each other because of  $J_2$ , making  $|\uparrow\downarrow\uparrow\downarrow \dots\rangle$  and  $|\downarrow\uparrow\downarrow\uparrow \dots\rangle$  the most probable configurations. Therefore, the system is in the nearest neighbor antiferromagnetic phase.
2. when  $J_1 > 0$  and  $J_2 \gg J_1/2$  and both  $J_1$  and  $J_2$  dominate  $h$ , the nearest and next-nearest neighbors want to align parallel to each other, making  $|\uparrow\uparrow \dots \uparrow\rangle$  and  $|\downarrow\downarrow \dots \downarrow\rangle$  as the most probable configurations. Therefore, the system is in the ferromagnetic phase.
3. when  $J_2 \ll 0$  and  $|J_2| > |J_1|/2$  and both  $J_1$  and  $J_2$  dominate  $h$ , the next-nearest neighbors want to be aligned anti-parallel to each other because of  $J_2$ , while the nearest neighbors want to be aligned parallel (if  $J_1 > 0$ ) or anti-parallel (if  $J_1 < 0$ ). This makes  $|\uparrow\uparrow\downarrow\downarrow \dots\rangle$ ,  $|\downarrow\downarrow\uparrow\uparrow \dots\rangle$ ,  $|\downarrow\uparrow\uparrow\downarrow \dots\rangle$  and  $|\uparrow\downarrow\downarrow\uparrow \dots\rangle$  the most probable configurations. Therefore, the system is in the next-nearest neighbour antiferromagnetic phase.

However, since we are studying a system with open boundary conditions, the ground state of the system in the next-nearest neighbour antiferromagnetic phase depends on the sign of  $J_1$ . When  $J_1 < 0$ , the two most probable states in the ground state are  $|\uparrow\downarrow\uparrow\downarrow\dots\rangle$  and  $|\downarrow\uparrow\downarrow\uparrow\dots\rangle$ . When  $J_1 > 0$ , the two most probable states in the ground state are  $|\uparrow\uparrow\downarrow\downarrow\dots\rangle$  and  $|\downarrow\downarrow\uparrow\uparrow\dots\rangle$ . When  $J_1 = 0$ , the four states are the most probable and equiprobable.

The system is said to exhibit ‘frustration’ when  $J_2 < 0$  and  $J_1$  is non-zero. When these conditions hold, the nearest and next-nearest neighbor interactions compete with each other, causing the spin at the site in consideration to be ‘frustrated’ with regard to its alignment ( $\uparrow$  or  $\downarrow$ ). In this work, we restrict ourselves to both the 1- $d$  Ising models described above, with  $h = 1$  and open boundary conditions.

## 2.2 Machine Learning and Neural Networks

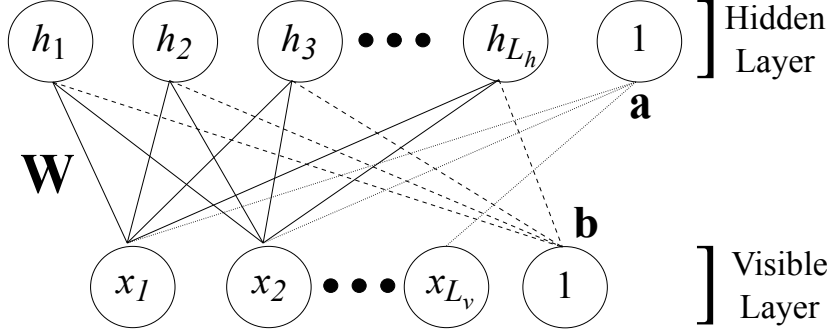
Machine Learning, as a sub-field of Artificial Intelligence strives to equip machines with the ability to learn from data or experiences (using statistical models or techniques), without being explicitly programmed to do so. In most scenarios, we need data to learn from. Given the data, we create a *machine learning model*, which has some parameters that help transform said data into relevant output. The key idea is to arrive at a model that learns useful representations from the data. For learning, we *train* the model by employing an optimisation algorithm to arrive at the best choice of parameters that accomplish a certain goal given by a chosen *objective function*. The model can then be used for inference or prediction.

While there are three main classes of learning in machine learning: supervised, unsupervised and reinforcement learning, in this thesis we focus only on unsupervised learning. In unsupervised learning, the data provided to the training process has no labels. Therefore it tries to find interesting representations within the data, without any help from the labels. This type of learning commonly involves tasks such as clustering, anomaly detection and dimensionality reduction. In the following subsections, we give a brief overview of the two neural network architecture types used in this thesis – restricted Boltzmann machine (RBM) and multilayer perceptron (MLP).

### 2.2.1 Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) is an energy based model, commonly used in unsupervised learning and as a generative model. Therefore, it is used to learn a probability distribution  $p(\mathbf{x})$  from given input  $\mathbf{x}$ . Additionally, RBMs can also be used to approximate the distribution  $\hat{p}(\mathbf{x}; \theta)$  where the probability distribution depends on some parameters  $\theta$ .

A restricted Boltzmann machine is an undirected probabilistic graphical model, that can be composed as a shallow neural network with two layers, a visible (input) layer and a hidden



**Figure 2.2:** The architecture of a restricted Boltzmann machine with  $L_v$  visible nodes and  $L_h$  hidden nodes. The solid lines denote the weights  $\mathbf{W}$ , the dotted lines denote the visible biases  $\mathbf{a}$  and the dashed lines denote the hidden biases  $\mathbf{b}$ .

layer as shown in Figure 2.2. Each of the  $N$  nodes in the visible layer represent the value of an input. Consequently, the only design choice is the choice of number of hidden variables. Typically, a multiple of the number of visible nodes, denoted by  $\alpha$ , is considered. The hidden layer, therefore, consists of  $\alpha \times N$  nodes. Thus the weight matrix for a restricted Boltzmann machine is fully described by the  $N \times (\alpha \times N)$  matrix of weights, where each weight  $W_{i,j}$  denotes the connection between the visible node  $x_i$  and hidden node  $h_j$ . There are also bias terms in the visible as well as hidden layers, denoted by  $a$  and  $b$ , respectively. Nodes in  $\mathbf{x}$  as well as those in  $\mathbf{h}$  take on binary values, specifically  $-1, 1$  for our context. The parameters  $W$ ,  $a$  and  $b$  are learnable parameters.

The restricted Boltzmann machine represents the joint probability distribution  $p$  for each node in the hidden and visible layers given by the Boltzmann distribution given in Eq. 2.7 where  $E(x, h)$  is the energy function and  $Z_w = \sum_{x,h} \exp^{-E(x,h)}$  is the normalisation partition constant.

$$p(x, h) = \frac{\exp^{-E(x,h)}}{Z_w} \quad (2.7)$$

The energy function for the restricted Boltzmann machine is given by Eq. 2.8

$$\begin{aligned} E(\mathbf{xh}) &= -\mathbf{a}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} - \mathbf{x}^\top \mathbf{W} \mathbf{h} \\ &= -\sum_{i=1}^N a_i x_i - \sum_{j=1}^H b_j h_j - \sum_{i=1}^N \sum_{j=1}^H W_{ij} h_j x_i \end{aligned} \quad (2.8)$$

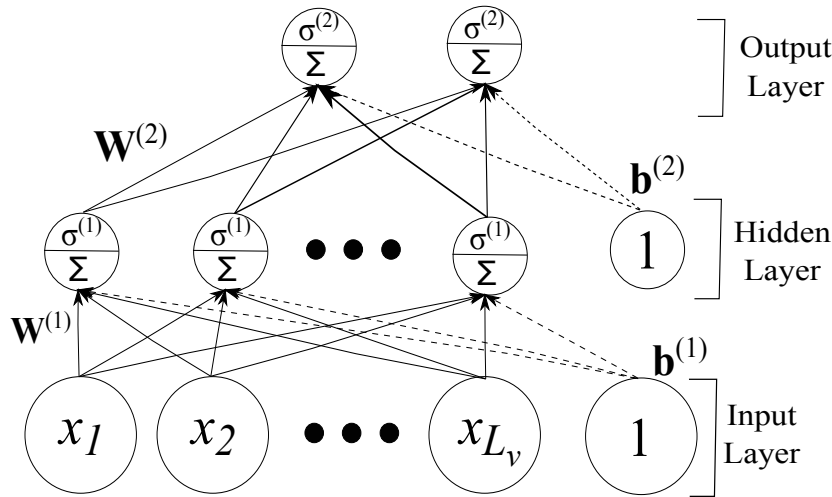
In case of the RBM, we are interested in finding the distribution of the input which can be

expressed as the marginal probability  $p(x)$  given in Eq. 2.9.

$$p(\mathbf{x}) = \frac{1}{Z_W} \exp\left\{(\mathbf{a}^\top \mathbf{x})\right\} \prod_i 2 \cosh\left(\sum_j x_j W_{ji} + b_i\right) \quad (2.9)$$

RBMs can be trained in supervised or unsupervised manner. In the supervised setting, they are usually used as feature extractors, but they can also be used for classification and regression tasks as in [51, 56, 61, 65, 87]. In the unsupervised setting, they have been used in a variety of domains such as face recognition [85], dimensionality reduction [39], unsupervised feature learning [20] and image denoising [84], topic modelling [40, 97], acoustic modelling [24, 41], collaborative filtering [80], anomaly detection [30], fault detection [53] and credit scoring [87].

### 2.2.2 Multilayer Perceptron



**Figure 2.3:** The architecture of a multilayer perceptron with  $L_v$  visible nodes and one hidden layer. The visible layer consists of visible nodes  $x_1, \dots, x_{L_v}$ . The connections between any two consecutive layers are given by the weight matrix  $\mathbf{W}$ . The solid lines denote the weights  $\mathbf{W}$ , while the dashed lines denote the biases  $\mathbf{b}$ .

Multilayer Perceptrons (MLP), as the name implies, are composed of an input layer to receive the signal, an output layer that makes a decision or prediction depending on the task at hand and the input; and in between these two are arbitrary number of hidden layers (Figure 2.3). Therefore, in the case of an MLP, once the input has been processed by the hidden layer it is evaluated against either a loss function in case of supervised learning, or an objective function in an unsupervised setting, at the output layer. The result is then backpropagated to update the weights, then fed forward again. This process continues until the error is sufficiently reduced or the objective function minimized to some threshold.

A multilayer perceptron is associated with a directed acyclic graph that can be composed into several layers, representing a composition of functions from every layer,  $\hat{f}(\mathbf{x}, \theta) = f^{(l)}(\dots f^{(2)}(f^{(1)}(\mathbf{x}; \theta^{(1)}); \theta^{(2)}) \dots; \theta^{(l)})$ , where  $l$  denotes the total number of layers and  $f$  is a vector function. Each node in one layer is fully connected with a certain weight to every node in the following layer. Thus, the weights at layer  $l$  can be represented by a matrix  $\mathbf{W}^{(l)}$  such that  $\mathbf{W}_{i,j}^{(l)}$  represents the weight from node  $i$  in layer  $l-1$  to node  $j$  in layer  $l$ . In the input and hidden layers, there are the special nodes called the bias nodes, which always receive 1 as an input. Each of these bias nodes is connected to the nodes in the subsequent layer, except the bias node in that layer. The weight of this particular connection is called *bias*. The number of hidden layers  $N_l$  and the number of hidden nodes in a layer  $N_h^{(l)}$  determine the expressiveness of the neural network (and the function  $\hat{f}$ ), and are tuned as hyperparameters to find the best values for the problem at hand.

The output of a node  $i$  in layer  $l$  is given by  $\hat{f}_i^{(l)} = \sigma^{(l)}(\mathbf{W}_{:,i}^{(l)} f^{(l-1)} + b_i^{(l)})$ , where  $\sigma^{(l)}$  is the activation function employed at layer  $l$  and  $f^{(l-1)}$  is the output vector from the previous layer, and  $f^{(0)} = \mathbf{x}$ , the input. In terms of a composition of functions, a multilayer perceptron with  $l$  layers computes  $\hat{f}(\mathbf{x}; \theta)$  as in Eq. 2.10, where  $\theta = \theta^{(1)}, \dots, \theta^{(l)} = W^{(1)}, b^{(1)}, \dots, W^{(l)}, b^{(l)}$ .

$$\begin{aligned} \hat{f}(\mathbf{x}; \theta) &= f^{(l)}(\dots f^{(2)}(f^{(1)}(\mathbf{x}; \theta^{(1)}); \theta^{(2)}) \dots; \theta^{(l)}) \\ &= \sigma^{(l)}(\mathbf{W}^{(l)} \dots \sigma^{(2)}(\mathbf{W}^{(2)} \sigma^{(1)}(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots + \mathbf{b}^{(l)}) \end{aligned} \quad (2.10)$$

Training of the multilayer perceptron strives to find  $\theta$  such that  $\hat{f}$  approximates the function underlying the given data. Therefore, we need to specify some loss or objective function  $\mathcal{L}(\theta)$  to quantify how well or poorly it is able to do so. Mean-squared error and cross-entropy are some commonly used loss functions. After choosing a loss or objective function, iterative gradient descent algorithm is used to find the best  $\theta$  by minimising said loss or objective function. Some variants of gradient descent (such as minibatch stochastic gradient descent) and optimiser algorithms (such as RMSProp and Adam) are commonly used to achieve better convergence.

Multilayer perceptrons have been applied to problems where the input data needs to be classified into multiple labels [70, 63], across various domains. This is also true for regression problems where a prediction is desired from the neural network, as in [22, 33]. MLPs have also been found successful in applications across text mining [23, 46], feature selection [76], image recognition [36], and several other problem areas as outlined in surveys across different fields [91, 48, 54].

In summary, RBM is a generative model and is used to model probability distributions. In contrast, the MLP is a discriminative model which strives to approximate a function that best fits the given data. Despite their differences, each of them can be carefully designed to address the applications typically suited to the other.



### 2.2.3 Transfer Learning

Transfer learning proposes to reuse a machine learning model that was trained for a particular task, to initialise another machine learning model that will subsequently be trained to perform another, related but different, task. This approach has shown efficiency and effectiveness improvements in neural networks [99].

When employing transfer learning, we first train a neural network for a task which we call the *base task*. This neural network corresponding to the base task is called the *base network*. Next, we want to train another neural network for another similar task, which we call the *target task*. This neural network which we train for the target task is called the *target network*. For the transfer, we initialise the target network with the parameters of the base network, instead of a random set. The former is called a *hot-start* while the latter (*i.e.*, randomly initialised) is called a *cold-start*. After the transfer, the target network is further trained and this step is called *fine-tuning* the network for the target task.

Note that there are several issues that arise when naively transferring parameters from one network to another. Typically, the base and target networks have the same architecture, leaving no flexibility in changing the structure of the target network. In case the input dimensions do not match for the two networks, we project or embed the input of the target task such that its dimensions match that of the base task (such as image resizing in computer vision tasks). Further, if the output of the two networks differ in their dimensions, we avoid transferring the output layer; transferring only several weights close to the input layer and initialising the other layers randomly. Freezing of several layers, *i.e.*, not training them, close to the input is another strategy often used to improve the efficiency of training very deep neural networks.

Transfer learning has been applied to several unsupervised, supervised and reinforcement learning tasks [71, 93]. It has also been applied to restricted Boltzmann machines for tasks such as reinforcement learning [4] and classification [92, 103]. The authors of [99] applied transfer learning to deep convolutional neural networks and found that the transferred features produced improved generalisation, that lasted even after fine-tuning to the target dataset.

In this work, we employ transfer learning across system sizes (and parameters) to determine ground state energy for the Ising  $J_1 - J_2$  model.

## 2.3 Neural-network Quantum States (NQS)

Carleo and Troyer in [12] demonstrate that Artificial Neural Networks (ANNs) are capable of capturing the intricacies of quantum many body systems, using the Transverse Field Ising model as well as the Antiferromagnetic Heisenberg (AFH) model. Their scheme falls under the family of variational quantum Monte Carlo methods, which uses neural networks as the

trial wave function. The objective in this approach is to represent a quantum state  $\Psi(\mathbf{x}, \theta)$  using a neural network.

The neural network quantum states method has also been explored to study quantum entanglement properties [26], its relation to other approaches [16, 35] and its expressiveness [21, 57, 43, 32]. NQS have also been used to find excited states [17, 62], and this was also explored as part of this work for the Ising model.

The wavefunction  $|\Psi\rangle$  of a quantum system can be represented as,

$$|\Psi\rangle = \sum_x \Psi(x)|x\rangle, \quad (2.11)$$

where  $|x\rangle$  is the basis state vector associated with the configuration  $x$  and  $\Psi(x)$  is the complex coefficient representing the probability amplitude for  $x$ .

The probability of a configuration  $x$  occurring in the state  $|\Psi\rangle$ , is given by

$$p_x = \frac{|\Psi(x)|^2}{\sum_x |\Psi(x)|^2} = \frac{|\Psi(x)|^2}{Z_\Psi} \quad (2.12)$$

where  $\sum_x |\Psi(x)|^2 = Z_\Psi$ , defined as the normalisation constant.

The average energy of the system in a state  $|\Psi\rangle$  is defined in Eq. 2.13. The variational principle in quantum physics states that  $E[\Psi] \geq E_0$  for all  $|\Psi\rangle$ s, where  $E_0$  is the **ground energy** of the system. The problem of finding the ground state energy thus reduces to the problem of minimizing  $E[\Psi]$ .

$$E[\Psi] \equiv \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} \quad (2.13)$$

Interestingly, Eq. 2.13 is similar to the Rayleigh quotient,  $R(M, x) = \frac{x^* M x}{x^* x}$  defined for a given complex Hermitian matrix  $M$  and non-zero vector  $x$ . It can be shown that for a given matrix, the Rayleigh quotient reaches its minimum value (i.e., the smallest eigenvalue of  $M$ ) when the corresponding eigenvector is also at its minimum. Further, the Rayleigh quotient gives the expectation value of the observable corresponding to the operator  $M$  for a system whose state is given by  $x$ . For Eq. 2.13, this implies that the various eigenvalues of  $H$ , arranged in increasing order, represent the energy levels of the system. Additionally, the eigenvectors corresponding to the ordered eigenvalues map to the various excited energy states. We exploit this idea to determine excited states and detail the same in Chapter 4.

Eq. 2.14 is obtained on performing algebraic transformations<sup>2</sup> on  $E[\Psi]$ , where  $E_{loc}(x)$  is the local energy of a configuration  $x$ , and is defined in Eq. 2.15.

---

<sup>2</sup>See 6.2 for details

$$E[\Psi] = \sum_{x \in \mathbf{x}} \frac{|\Psi(x)|^2}{Z_\Psi} E_{loc}(x) \quad (2.14)$$

$$E_{loc}(x) = \sum_{x' \in \mathbf{x}'} H_{x,x'} \frac{\Psi(x')}{\Psi(x)} \quad (2.15)$$

where,  $H_{x',x} = \langle x|H|x' \rangle$ , the entry of the Hamiltonian matrix for configurations  $x$  and  $x'$

Note that the expression for  $E[\Psi]$  is a weighted average over  $E_{loc}(x)$ , and that the number of configurations,  $\mathbf{x}$ , grows exponentially with the size of the system. Therefore, we use Monte Carlo methods to sample  $x$  from the probability distribution  $\frac{|\Psi(x)|^2}{Z_\Psi}$ , using which the expectation of the local energy  $E_{loc}(x)$  is calculated, to arrive at the value of  $E[\Psi]$ .

$$E[\Psi] = \mathbb{E}_{\mathbf{x} \sim \frac{|\Psi(x)|^2}{Z_\Psi}} [E_{loc}(x)] \quad (2.16)$$

In this work, we use both MLP and RBM architectures for finding the ground state and a complex implementation of the MLP for finding the first few excited states. The following subsections go into details about the specifics of the architectures employed.

### 2.3.1 Restricted Boltzmann Machine (RBM) for NQS

[101] demonstrates the feasibility of RBMs for finding the quantum critical points of quantum many-body systems represented by the Ising model. RBM is a generative model, which makes it ideal for learning the underlying probability distribution  $p(\mathbf{x})$  for a given input  $\mathbf{x}$ . Mathematically, the probability distribution  $p(\mathbf{x})$  from an RBM is given by

$$p_{RBM}(\mathbf{x}) = \frac{1}{Z_{RBM}} \exp\{(\mathbf{a}^\top \mathbf{x})\} \prod_i 2 \cosh \left( \sum_j x_j W_{ji} + b_i \right) \quad (2.17)$$

where  $Z_{RBM}$  is the normalization constant summed over all possible configurations;  $\mathbf{a}$ ,  $\mathbf{b}$  are biases associated with the visible and hidden layers, respectively; and  $W$  is the weight matrix

As Eq. 2.12 gives the probability of occurrence of a given configuration,  $\mathbf{x}$ , in the state  $\Psi(x)$ , equating it with  $p_{RBM}$  from above gives us,

$$\begin{aligned} \frac{|\Psi(x)|^2}{Z_\Psi} &= p_{RBM}(\mathbf{x}) \\ \implies \Psi(x) &= \sqrt{\frac{Z_\Psi}{Z_{RBM}} \exp\{(\mathbf{a}^\top \mathbf{x})\} \prod_i 2 \cosh \left( \sum_j x_j W_{ji} + b_i \right)} \end{aligned} \quad (2.18)$$

Next, on substituting  $\Psi(x)$  from Eq. 2.18 into Eq. 2.15,  $E_{loc}(\mathbf{x})$  becomes:

$$E_{loc}(\mathbf{x}) = \sum_{\mathbf{x}'} H_{\mathbf{x}, \mathbf{x}'} \exp \left\{ \left( \frac{1}{2} \mathbf{a}^\top (\mathbf{x}' - \mathbf{x}) \right) \right\} \sqrt{\prod_i \frac{\cosh \left( \sum_j x'_j W_{ji} + b_i \right)}{\cosh \left( \sum_j x_j W_{ji} + b_i \right)}} \quad (2.19)$$

The RBM is trained in an unsupervised manner, relying on the random configurations it generates. The minimisation process steps through Gibbs sampling of configurations, the calculation of the expected value of the local energy for sampled configurations and stochastic gradient descent (specifically, RMSProp) until a predefined stopping criterion is met.

### 2.3.2 Multilayer Perceptron (MLP) for NQS

Cai and Liu [9] use a multilayer perceptron to represent the wavefunction for quantum many-body systems. For an MLP with one hidden layer, the output is given by

$$o(\mathbf{x}) = \sigma_2(W_2\sigma_1(W_1\mathbf{x} + b_1) + b_2) \quad (2.20)$$

where  $W_1$  and  $W_2$  are the weight matrices from the input to the hidden layer and from the hidden to the output layer, respectively;  $\sigma_1$  is the activation function applied to the nodes in the hidden layer; and  $b_1$  and  $b_2$  are the biases for the input and the hidden layers, respectively. An activation function may be applied to the output layer too, depending on the task at hand. For instance, the activation function  $\sigma_2$  is applied to the output layer to restrict the output to only positive real numbers, in the case of the MLP employed for determining the ground state.

In the context of the quantum many-body problem, the input  $x$  for the MLP refers to the system's configuration. The output layer consists of only one node which represents  $\Psi(x)$ , with the ReLU activation function in order to restrict it to real and positive wavefunctions, when finding the ground state. Since the output of the MLP represents the wavefunction,  $\Psi(x)$  for the MLP is given by,

$$\Psi(\mathbf{x}) = o(\mathbf{x}) = \sigma_2(W_2\sigma_1(W_1\mathbf{x} + b_1) + b_2)$$

As with the RBM, substituting  $\Psi(x)$  from the above equation into Eq. 2.15,  $E_{loc}(\mathbf{x})$  becomes,

$$\implies E_{loc}(\mathbf{x}) = \sum_{\mathbf{x}'} H_{\mathbf{x}, \mathbf{x}'} \frac{\sigma_1(W_2\sigma_2(W_1\mathbf{x}' + b_1) + b_2)}{\sigma_1(W_2\sigma_2(W_1\mathbf{x} + b_1) + b_2)} \quad (2.21)$$

As opposed to the RBM, the benefit of employing the MLP architecture is that it allows one to model non-binary spins, aside from the previously mentioned fact that it can be trained in a supervised manner. Note that in this thesis, we restrict ourselves to binary spins, even when we evaluate the MLP architecture.

On the other hand, an advantage of employing RBMs is that the samples can then be drawn using Gibbs sampling which is more efficient. Note that this can only be done for real and positive wavefunctions. In contrast, we can only use Metropolis sampling for drawing samples when using the MLP architecture, for any wavefunction.

## 2.4 The NQS library

There exists the **NetKet** [10] library which is an open-source project for the study of quantum many-body systems with machine learning techniques employing neural quantum states. As of this writing, the library is implemented in **Python** atop the **JAX** [8] framework and includes modules supporting Exact Diagonalisation, variational Monte Carlo and Matrix Product States methods, for supervised as well as unsupervised learning settings. The library supports Message Passing Interface (MPI) for distributed and parallel computing.

While distributed and parallel computing have their merits, recent advances in machine learning were partly due to the use of general purpose graphics processing units (GPUs), which enable significant speedups at a reasonable hardware cost. GPUs typically have more cores than central processing units (CPUs), even though the former are usually slower. As a result, GPUs are suitable for parallelising vector and matrix operations, and that is why they're commonly used for high performance computer graphics. As neural network operations involve lots of matrix operations, GPUs make a fitting case for such models. The caveat with GPUs is that they have a lower memory capacity but higher memory latency than CPUs. However, they have higher memory bandwidth than CPUs. Note that for conducting parallel or distributed computing jobs, one needs several processors or computers, respectively. On the other hand, GPU based parallelism for speedup in matrix operations can be done with a single computer that has a general purpose GPU. The latter option is generally economical than the former.

Further, many of the popular machine learning frameworks, such as **TensorFlow** [2] and **PyTorch** [72], facilitate building and running neural network operations on top of GPUs without having to interface directly with the GPUs application programming interface (such as **CUDA** or **OpenCL**). This significantly reduces the barrier to entry, at least in terms of infrastructure, for the general public to build and try out machine learning models.

It was for these reasons that the library **nqs-tensorflow2** [100] was designed, to provide neural quantum states operations with **TensorFlow**.

All implementations described in this thesis, make use of the modules provided in this library and can be readily deployed on general purpose GPUs. The lattices we model for the different system sizes leverage the **Hypercube** class in the **Graph** package. The Ising models we use come from the **Ising** and **IsingJ1J2** classes within the **Hamiltonian** package. Similarly, the machine learning models we use in our experiments come from the **Model**

package, which is further split into packages including the `mlp.complex`, `mlp.realpos` and `rbm.realpos` packages. The experiments covered in this thesis are all in unsupervised settings and use the `Learner` class in the `learner` package. Lastly, the work in this thesis employs three variants of the Metropolis-Hastings algorithm, which is a Markov Chain Monte Carlo (MCMC) sampling method. The implementation for these resides in the `Sampler` package as the `Gibbs`, `MetropolisAll` and `MetropolisLocal` classes. Note that in the `MetropolisLocal` variant of the Metropolis algorithm, new configurations are proposed by flipping random spins in the current configurations. When working with sparse matrices, such as the Hamiltonian for the ground state, this method is more efficient. On the other hand, the `MetropolisAll` implementation proposes a new configuration by sampling from the uniform random distribution, and is applicable to Hamiltonian matrices in general. Additionally, the `Logger` class is used to collate results at the end of experiments for visualisation in the form of plots shown in Chapters 3 and 4.

## 2.5 Related Work

Machine learning in general, and neural networks, in particular, have proven their mettle in solving different problems across various domains. Scientific machine learning [6] is an emerging interdisciplinary field that leverages machine learning for problems in engineering and science domains. Within this domain of scientific machine learning, there exists the sub-field of machine learning for quantum physics, which is the focus of this thesis. Quantum machine learning [3, 27] studies the connection between quantum physics and machine learning. Recently, classical machine learning algorithms, specifically neural networks proved fruitful in identifying phases of quantum matter [14], performing quantum tomography [15], representing quantum systems [12], devising error correction protocols [88] and many other applications. For more examples and a detailed review, we refer the reader to [13, 81]. For specific applications with restricted Boltzmann machine, one should see [60, 89].

The authors of [50] were the first to use a neural network to represent a quantum state, using a multilayer perceptron to do so by solving the Schrödinger equation. In this work, the Schrödinger equation is used as the loss function for the multilayer perceptron. We instead focus on the more recent and seminal work by Carleo and Troyer [12], which the authors refer to as neural-network quantum states. A brief overview of neural-network quantum states is available in [11, 13, 42] for those interested.

The work by Carleo and Troyer in [12] used restricted Boltzmann machines to represent and simulate quantum many-body systems and this method falls into the family of variational quantum Monte Carlo methods. The authors tested their approach on the Ising and Heisenberg models, demonstrating that neural networks were indeed capable of finding the ground state and of reproducing the time evolution of these systems to study their dynamics. Even though this work leveraged restricted Boltzmann machines, and defined neural-network

quantum states to be associated with that architecture, the body of literature it generated evolved the term into an umbrella term that solves quantum physics problems with any neural network architecture.

Most of the work that evaluates the effectiveness of restricted Boltzmann machine architecture of the neural-network quantum states does so in finding the ground state of different quantum systems. It has been used to study the Ising model [12, 96], Heisenberg model [12, 68], Heisenberg  $J_1 - J_2$  model [90, 94, 68], Holstein model [66], Hubbard model [68], fermions [18, 73], and topological states [25, 35]. There are several works that also evaluate excited states. The authors of [17] used restricted Boltzmann machine neural-network quantum states to determine low-lying excited states in the Heisenberg and Bose-Hubbard models, leveraging imaginary time evolution with stochastic reconfiguration from the ground state. The authors of [67, 69] find excited states of the Heisenberg  $J_1 - J_2$  model by optimising the neural-network quantum states for different quantum number sectors, while the authors of [37, 90] did a direct optimisation of the excited states.

Unsupervised multilayer perceptron neural-network quantum states are mainly used to determine the ground state of several quantum systems. They have been used to find the ground state of Bose-Hubbard models with bosons on a lattice [77, 79] and on a continuous space [78], Hubbard model [58] and Heisenberg model [9]. A variant of the multilayer perceptron, the radial basis function network, has also been used by the author of [86] to determine ground states of two quantum mechanical systems— harmonic oscillator and particle in a box. The authors of [9] used a supervised MLP neural-network quantum states to find the ground state of free bosons, free fermions, and antiferromagnetic Heisenberg  $J_1 - J_2$  model. The authors of [94] analysed the generalisation capabilities of neural-network quantum states by finding the ground state of the Heisenberg  $J_1 - J_2$  model in a supervised manner. Their experiments showed that the generalisation ability of the neural-network drops as frustration is increased. In finding the excited states, unsupervised multilayer perceptron neural-network quantum states were also used by the authors of [17] for the Heisenberg and Bose-Hubbard model. The author of [62] finds the excited states for a quantum harmonic oscillator by adding orthogonal constraints to the ground state in the optimisation process. One of our approaches to find excited states in the Ising model derives from this work, as detailed in Sec. 4.1.1.

Additionally, the authors of [17] also compare the multilayer perceptron against restricted Boltzmann machine, wherein they found that a single hidden layer MLP has comparable performance to the RBM. However, they also observed that three-layered MLP is more effective than the RBM. Nonetheless, they further state that this doesn't strictly imply that the former are better than the latter. All of the work cited above used different variations of multilayer perceptron, in terms of the number of hidden layers and hidden nodes needed.

The works that use neural-network quantum states have also employed architectures such as

convolutional neural networks (CNN) [19, 79], recurrent neural networks (RNN) [38, 83] and graph neural networks (GNN) [98], of which the latter is the work with possibly the highest number of particles simulated with neural-network quantum states.

The list given above is by no means an exhaustive one but it establishes that the focus of existing work has been on studying the effectiveness of neural-network quantum states for different quantum many-body problems. In this work however, we focus on improving the scalability, efficiency, while improving or enhancing the effectiveness, when finding the ground state energy of the quantum systems. Also, we restrict ourselves to prototypical models, i.e., Ising and Ising  $J_1 - J_2$ , in this thesis. To the best of our knowledge, the latter hasn't been studied with neural-network quantum states, despite it having a frustration phenomenon (although not as complex as that evident in the Heisenberg  $J_1 - J_2$  model).

The contributions made in this thesis, borrow heavily from the work cited above, particularly the transfer learning protocols proposed by Zen et al. in [102] in the context of the first contribution. This thesis extends their evaluation to the Ising  $J_1 - J_2$  model using RBM and MLP neural-network quantum states. In light of finding excited states (the second contribution) for the Ising model, one of the methods (Sec. 4.1.1) evaluated in Chapter 4 is derived from the one proposed in [62], but applied to the Ising model.



# CHAPTER 3

## Ground State Energy for Ising $J_1 - J_2$ model using NQS

In this chapter we describe the physics-inspired transfer learning protocols we leverage to find the ground state energy. These protocols help improve the scalability, efficiency and effectiveness of the RBM and MLP based NQS architectures, when determining the ground state. The rest of this chapter is organised as follows. The following section introduces transfer learning and Sec. 3.1.1 gives an overview of the transfer learning protocols we used in finding the ground state. Section 3.2 details the performance evaluation and results from these protocols. Finally, Section 3.3 summarises and concludes this chapter.

### 3.1 Transfer Learning for finding Ground State Energy

As described in Section 2.3, the objective function we minimize to determine the ground state is given by:

$$E[\Psi] = \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} = \mathbb{E}_{\mathbf{x} \sim \frac{|\Psi(\mathbf{x})|^2}{Z_\Psi}} [E_{loc}(\mathbf{x})] \quad (3.1)$$

For the minimization of  $E[\Psi]$ , we use Stochastic Gradient Descent and the gradient of  $E[\Psi]$  is given by Eq. 3.2. The reader is referred to Section 3.1 of [55] for details on the same.

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= E[\Psi] = \mathbb{E}_{\mathbf{x} \sim \frac{|\Psi(\mathbf{x})|^2}{Z_\Psi}} [E_{loc}(\mathbf{x})] = \sum_{\mathbf{x}} p(\mathbf{x}, \boldsymbol{\theta}) E_{loc}(\mathbf{x}, \boldsymbol{\theta}) \\ \implies \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \mathbb{E}[E_{loc} D_{\boldsymbol{\theta}}^*] + \mathbb{E}[E_{loc}^* D_{\boldsymbol{\theta}}] - (\mathbb{E}[E_{loc}] \mathbb{E}[D_{\boldsymbol{\theta}}] + \mathbb{E}[E_{loc}^*] \mathbb{E}[D_{\boldsymbol{\theta}}^*]) \\ &= 2 \operatorname{Re} (\mathbb{E}[E_{loc} D_{\boldsymbol{\theta}}^*] - \mathbb{E}[E_{loc}] \mathbb{E}[D_{\boldsymbol{\theta}}^*]) \end{aligned} \quad (3.2)$$

where

$$D_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\psi(\mathbf{x}, \boldsymbol{\theta})} \frac{\partial \psi}{\partial \boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{\theta})$$

Zen et al. evaluated the feasibility of NQS for finding the ground state energy of the simple Ising model in [101]. The objective function employed was the same as above, but the

Hamiltonian they used accounted only for the nearest neighbor interactions. While their evaluation included comparing transfer learning to cold-start runs, it only considered real positive RBMs for this purpose. In this work, we consider the Ising  $J_1 - J_2$  model and evaluate using RBM as well as MLP NQS architectures.

Recall that in the Ising  $J_1 - J_2$  model both nearest neighbor and next-nearest neighbor interactions are considered, resulting in the Hamiltonian given by Eq. 2.6. Accounting for the next nearest neighbor interactions only brings changes in the Hamiltonian matrix. Therefore, the objective function for this model stays the same as that described above, but with the Hamiltonian from Eq. 2.6.

In the next section, the transfer learning protocols evaluated for the Ising  $J_1 - J_2$  models as part of this thesis are described. These were proposed in [102] and are reproduced here for the sake of completeness.

### 3.1.1 Transfer Learning protocols for scalability

As mentioned previously in Sec. 2.2.3, the base and target networks may not have the same input dimensions. This is indeed the case, when we transfer weights from the neural network corresponding to a system of size  $N$  to that of size  $2N$ . Since the dimension of weights increases in the target network, we need to project the weights of the base network such that their dimension matches that required in the target network. In principle, this ‘projection’ can be done in a large number of ways but it wouldn’t be practical nor meaningful to consider all of them. Zen et al. [102] therefore formulate  $(k, p)$  – *tiling* transfer protocols, where groups of  $k$  weights calculated for a system of certain size are repeated  $p$  times as initialisation of weights for a system of larger size. Those different tiling protocols are explained below and diagrammatically shown in Fig. 3.1.

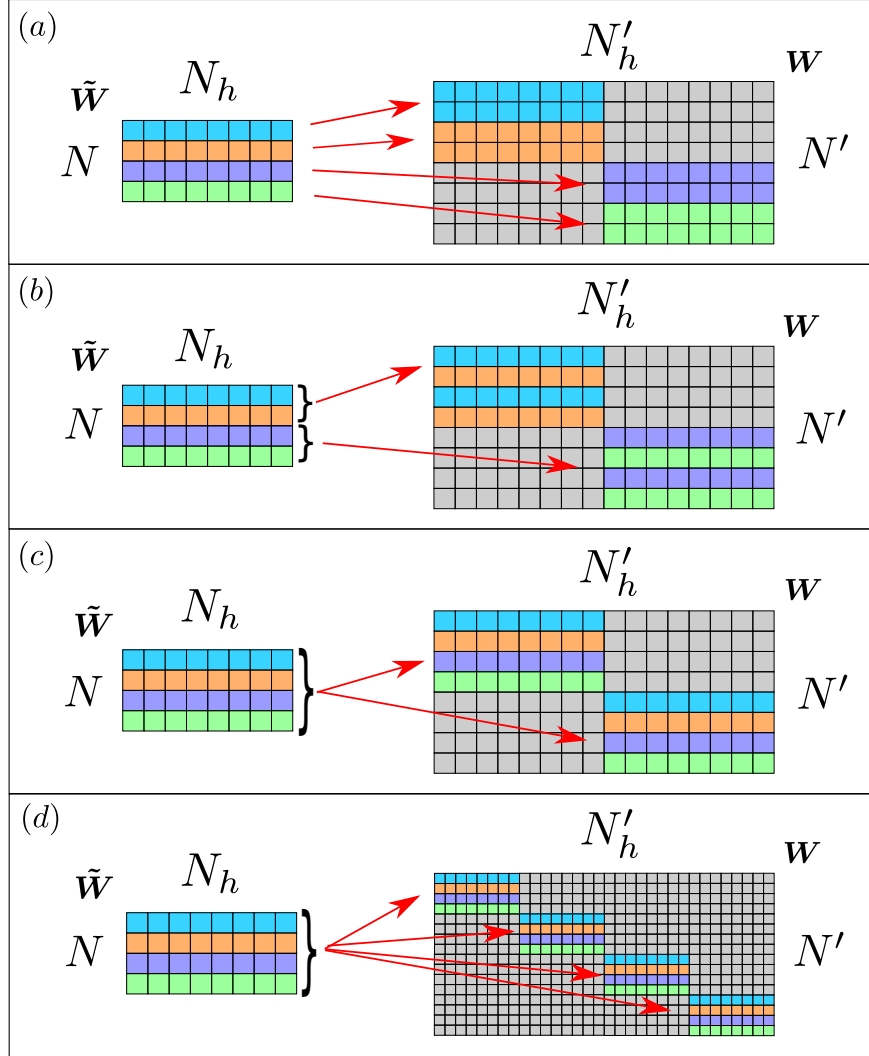
- $(1, 2)$  – *tiling* (Fig. 3.1(a)) Here  $p = 2$  and we double the weights as follows. The weights of the target network are initialised with  $W_{2j-1,i} = W_{2j,i} = \tilde{W}_{j,i}$  for  $j \in [1, N/2]$ ,  $W_{2j-1,i+N_h} = W_{2j,i+N_h} = \tilde{W}_{j,i}$  for  $j \in [N/2 + 1, N]$  and a random value for all the other terms. Observe that this tiling favors situations in which each site is equivalent, e.g., consider the state  $|\uparrow\uparrow\uparrow\uparrow\rangle$ ; this tiling will provide a bias towards the state  $|\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\rangle$  (spins in red are the ones ‘copied’ from the base network). This can be seen as favoring ferromagnetic correlations. On the other hand, if the base network represented an antiferromagnetic state,  $|\uparrow\downarrow\uparrow\downarrow\rangle$ , then the initialisation of the target network would be biased towards  $|\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\downarrow\rangle$ , which implies weaker antiferromagnetic correlations. Thus we expect this protocol to work well for paramagnetic or ferromagnetic phases, but not so well for a system in antiferromagnetic phase.
- $(2, 2)$  – *tiling* (Fig. 3.1(b)) This protocol is similar to the  $(1, 2)$  – *tiling* described above except that instead of doubling a single spin, we double a pair of them, because  $k = 2$ . The weights for the target system are given by  $W_{4j-3,i} = W_{4j-1,i} = \tilde{W}_{2j-1,i}$  and

$W_{4j-2,i} = W_{4j,i} = \tilde{W}_{2j,i}$  for  $j \in [1, N/4]$ , while for  $j \in [N/4 + 1, N/2]$ ,  $W_{4j-3,i+N_h} = W_{4j-1,i+N_h} = \tilde{W}_{2j-1,i}$  and  $W_{4j-2,i+N_h} = W_{4j,i+N_h} = \tilde{W}_{2j,i}$  and a random value for all the other terms. Consider a state  $|\uparrow\downarrow\uparrow\downarrow\rangle$ , for which this protocol would provide a bias towards the state  $|\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\rangle$  which preserves the nearest neighbor antiferromagnetic correlations in the base state. Additionally, for a next nearest neighbor antiferromagnetic state,  $|\uparrow\uparrow\downarrow\downarrow\rangle$ , this tiling would give a bias towards the state  $|\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow\rangle$  implying much weaker next-nearest neighbor antiferromagnetic correlations. Similarly, for a ferromagnetic state,  $|\uparrow\uparrow\uparrow\uparrow\rangle$  the protocol would give a bias towards the state  $|\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\rangle$ . Additionally, even for a zero magnetisation ferromagnetic state,  $|\uparrow\uparrow\downarrow\downarrow\rangle$ , the tiling would result in a bias towards the state  $|\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow\rangle$ , preserving the ferromagnetic correlation. Hence, we expect this protocol to work better for nearest neighbor antiferromagnetic phases compared to the  $(1, 2)$  – *tiling* protocol, as well as for ferromagnetic phases but not quite effective in next-nearest neighbor antiferromagnetic phases.

- $(L, p)$  – *tiling* (see Fig. 3.1(c) for the  $(L, 2)$  – *tiling* protocol and Fig. 3.1(d) for the  $(L, 4)$  – *tiling* protocol) By setting  $k = N$ , i.e., the size of the system, we transfer all the weights of the base network,  $\tilde{W}_{j,i}$ , on the first  $N$  visible nodes of the target network and then repeat them for the other  $(p - 1)N$  half on the visible layer nodes but coupled to the other hidden nodes. More formally, we match  $W_{j+\eta N, i+\eta N_h} = \tilde{W}_{j,i}$  for  $j \in [1, N]$ ,  $i \in [1, N_h]$  and  $\eta \in [0, p - 1]$ . We expect this protocol to be favorable in the antiferromagnetic phase because a state  $|\uparrow\downarrow\uparrow\downarrow\rangle$  would give a bias towards the state  $|\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\rangle$ . Similarly the state  $|\uparrow\uparrow\uparrow\uparrow\rangle$  would give a bias towards  $|\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\rangle$ , implying that this protocol would be favorable in the ferromagnetic phase as well. However, for a ferromagnetic state where the sum of spins is fixed to zero, the state  $|\uparrow\uparrow\downarrow\downarrow\rangle$  would give some bias towards  $|\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\downarrow\rangle$  which has weaker ferromagnetic correlations.

While these protocols may not be exhaustive, we see that they already provide some interesting insights into the efficiency and effectiveness of the transfer learning approach for the Ising  $J_1 - J_2$  model.

Note that aside from transferring weights between many-body quantum systems of different sizes, weights could also be transferred between systems of the same size, but different parameters. For instance, transferring weights from a system with  $J_2 = -0.9$  to another with  $J_2 = -1.0$ , both having the same number of particles (system size) and values for  $h$  and  $J_1$ . Transfer learning across parameters is useful for finding critical points and analysing phase transitions as in [101]. For evaluating transfer learning across parameters for the Ising  $J_1 - J_2$  model, we set  $h = 1$ ,  $J_1 = -0.5$  and varied  $J_2$  in the range  $[-3.0, 0.0]$  at increments of 0.1. We refer the reader to Sec. 6.3 in the appendix for details and some results on the same.



**Figure 3.1:** Schematic representation of the different transfer learning protocols used to scale up one-dimensional systems.  $N, N_h$  and  $N', N'_h$  correspond to the number of visible and hidden nodes for the base and target networks, respectively. The different panels show how to construct the target weight matrix  $\mathbf{W}$  from the base weight matrix  $\tilde{\mathbf{W}}$  by replicating the colored rows and filling the grey cells with random entries. Panels (a), (b) and (c):  $(1,2)$ -tiling,  $(2,2)$ -tiling and  $(L,2)$ -tiling, respectively. Here the scaling factor is 2. Panel (d) shows the  $(L,4)$ -tiling used when scaling up the network by a factor 4. Image and caption from [102].

### 3.2 Performance Evaluation

To evaluate the transfer learning protocols described above (proposed in [102]) in the context of the Ising  $J_1 - J_2$  model, we implemented them for RBM and MLP based NQS architectures in the unsupervised learning setting.

In order to account for the different phases of the one-dimensional Ising  $J_1 - J_2$  model, we

fixed  $h = 1$  and varied  $J_1$  and  $J_2$ , with  $J_1$  taking the values  $\{-0.5, 0.5\}$  and  $J_2$  taking the values  $\{-5.0, 5.0\}$ . In these experiments, we considered systems with  $N = \{8, 16, 32, 64, 128\}$  spins. The reported results are an average over 20 realisations of the same experiment, since our ground state calculation involves random components.

We evaluate both the *effectiveness* and *efficiency* for this approach. The effectiveness is reported as the mean relative error between the predicted and true energy, where the true energy was obtained from conventional methods (Exact Diagonalisation and MPS). The time needed to train the network is noted for evaluating the efficiency of the NQS employed to find ground states. This helps draw a comparison between the cold-start and transfer learning runs. These values are computed using the `time` module in Python, during isolated runs for each experiment, averaged over 20 iterations. Note that in the case of transfer learning, the true measure of convergence time needs to include the time taken to train all prior experiments from which the weights were transferred. For instance, for a system with  $N = 64$ , the time reported includes the time needed by a cold-start for  $N = 4$ , plus the time needed for hot-starts at  $N = 8, N = 16, N = 32$  and finally,  $N = 64$ . Thus, the time reported for the transfer learning experiments is the value accumulated over all prior experiments. For their cold-start counterparts, on the other hand, the individual time that the experiment took is reported.

As is typical with Neural Networks, there exist various choices for the optimiser (`RMSProp`, `Adam`, `SGD`), the activation function (`ReLU`, `tanh`, `sigmoid`), the learning rate (0.01, 0.001, 0.005), the number of hidden layers (1, 2 or 3) and the number of hidden nodes (as a multiple of number of visible nodes) for each of the experiments. The hyperparameters were chosen as a result of a grid search over these combinations.

We use an iterative optimisation process. At every iteration, we draw  $10^4$  samples to evaluate the energy  $\langle H \rangle_\Psi$  and its gradients using the parameters of the neural network (RBM or MLP). Next, we update the parameters of the neural network using a gradient descent algorithm. Specifically, we used the `RMSProp` optimiser to benefit from its adaptive learning strategies. The initial learning rate was set to 0.001. For the RBM, the samples were drawn from a single iteration of Gibbs sampling, while they were drawn using the Metropolis algorithm (specifically, the `MetropolisLocal` implementation) in case of the MLP. From our exploratory simulations we found choosing thrice as many hidden nodes as the visible ones to be the best effectiveness-efficiency tradeoff for the RBM. In case of the MLP, choosing `ReLU` as the activation function and 1 hidden layer with thrice as many hidden nodes as the number of visible nodes performed well. Additionally, during transfer learning and cold-starts, we need to set some parameters of the restricted Boltzmann machine randomly. To that end, the entries for the weight matrix  $\mathbf{W}$  were sampled from the Normal distribution,  $\mathcal{N}(0, 0.01)$ , i.e., with zero mean and standard deviation 0.01.

While training a dynamic stopping criterion was used to stop the simulations and this

criterion was based on the zero-variance principle in quantum Monte Carlo<sup>1</sup>. We set the threshold for standard deviation of local energy,  $E_{loc}$  to be 0.005, to keep it close to zero, in accordance with this principle. Additionally, the maximum number of epochs were set to  $3 \times 10^4$ , in case the standard deviation for  $E_{loc}$  never went below the threshold.

All experiments in this chapter were run on an NVIDIA DGX-1 server equipped with NVIDIA Tesla V100 graphics processing units with 460 tensor cores, 5120 CUDA cores, and 16GB memory on the infrastructure of Singapore National Supercomputing Centre [1].

For these experiments, the weights are first transferred from a quantum many-body system of size 4 to that of size 8, using the various tiling protocols, namely  $(L, 2)$ ,  $(1, 2)$ ,  $(2, 2)$  and  $(4, 2)$  as described in Sec. 3.1.1. Once the latter model converges to the minimal local energy, the learned weights are then transferred to a many-body system of size 16, again using all four of the tiling protocols. This process is then repeated for sequentially transferring weights to systems of size 32 from size 16, 64 from size 32 and 128 from size 64.

### 3.2.1 Ground State Energy using RBM

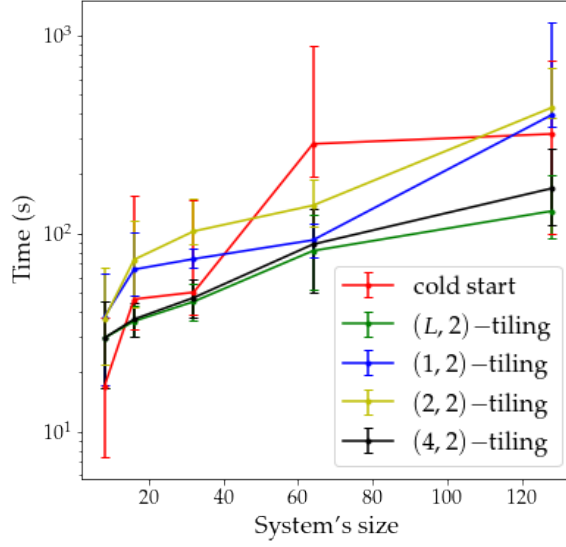
We first compare the efficiency of various transfer learning protocols and cold-starts for the Ising  $J_1 - J_2$  model across different values of  $J_1$  and  $J_2$ , indicating the various phases. Figures 3.2 to 3.5 plot the time to convergence for each of the systems we considered, as a function of its size. Each of these 4 figures shows results for the Ising  $J_1 - J_2$  model with  $(J_1 = -0.5, J_2 = -5.0)$ ,  $(J_1 = -0.5, J_2 = 5.0)$ ,  $(J_1 = 0.5, J_2 = -5.0)$ , and  $(J_1 = 0.5, J_2 = 5.0)$ , respectively. The vertical error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations. The shorter the error bar, the more consistent the results are across the iterations for that system size.

For cold-start, each system starts with a randomly initialised network, while for the transfer learning protocols, the initial parameters at size  $2N$  come from the parameters obtained at size  $N$ , when its simulation completes. Thus the compute time at size  $2N$  for the transfer learning case is given by the sum of, computation time for size  $N$  and computation time taken for size  $2N$  itself to meet the stopping criterion, starting from the transferred parameters. The sequence starts with a system of size  $N = 4$ , being randomly initialised with a cold-start.

In the next-nearest neighbor antiferromagnetic phases (Fig. 3.2 and Fig. 3.4), we see that  $(1, 2)$ -tiling and  $(2, 2)$ -tiling take the longest time to converge, even more than cold-start for some  $N$ . We also see that not only does  $(L, 2)$ -tiling have the lowest convergence time across system sizes among the transfer learning protocols, but its efficiency is also robust against the phase of the system. It is also evident that this difference in the convergence times for cold-start and  $(L, 2)$  tiling increases with system size.

---

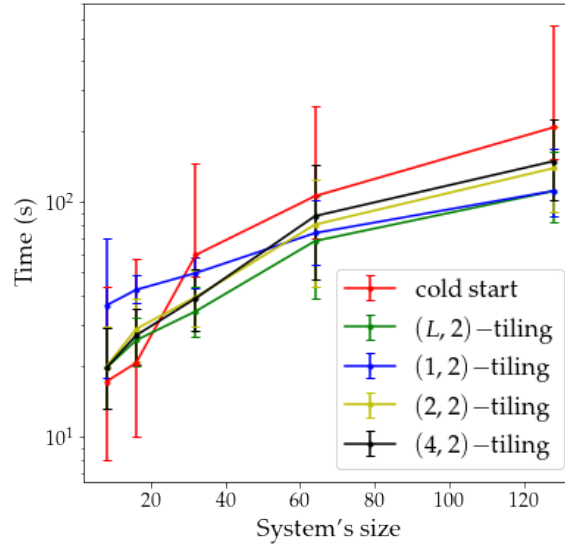
<sup>1</sup>i.e., the variance of local energy is exactly 0 if the given state is exactly the ground state wave function.



**Figure 3.2:** Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = -0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

In Figures 3.6 to 3.9, we plot the relative error  $\Delta E/E_0$  obtained across the different systems sizes- 8, 16, 32, 64, 128 for cold-start as well as the (1, 2), (2, 2), (4, 2) and (L, 2) tiling protocols, when the stopping criterion is reached. Each of these 4 figures shows results for the Ising  $J_1 - J_2$  model with  $(J_1 = -0.5, J_2 = -5.0)$ ,  $(J_1 = -0.5, J_2 = 5.0)$ ,  $(J_1 = 0.5, J_2 = -5.0)$ , and  $(J_1 = 0.5, J_2 = 5.0)$ , respectively. The vertical error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

We observe that (L, 2) – *tiling* is the most effective protocol, across sizes and phases. As expected (1, 2) – *tiling* performs poorly, often even worse than cold-start, in all the antiferromagnetic phases (Figures 3.6 to 3.8) and it matches the performance of the other tiling protocols in the ferromagnetic phase. The (2, 2) – *tiling* performs poorly in the next-nearest neighbor antiferromagnetic phase (Fig. 3.6 and Fig. 3.8) but performs comparably in the nearest neighbor antiferromagnetic phase (Fig. 3.7). This can be explained using an example configuration: given  $|\uparrow\uparrow\downarrow\downarrow\rangle$  in the next nearest antiferromagnetic phase, (2, 2) – *tiling* will give bias towards  $|\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\rangle$  which implies weaker next-nearest neighbor antiferromagnetic correlations. (L, 2) and (4, 2) tiling can be seen as the most effective protocols across the phases and sizes. Note that the bars for (L, 2) and (4, 2) tiling protocols are of the same length for size 8, since during the transfer from size 4 to size 8,  $L = 4$ , causing the tiling protocol to perform identically to (4, 2) tiling.



**Figure 3.3:** Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = -0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

### 3.2.2 Ground State Energy using MLP

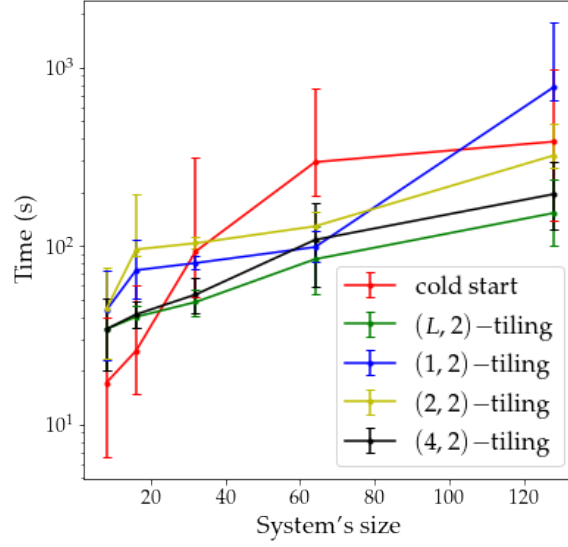
In this section we compare the efficiency of various transfer learning protocols and cold-start for the Ising  $J_1 - J_2$  model across the different phases with the MLP architecture.

For each cold-start run, the system starts with a randomly initialised network, while for the transfer learning protocols, the initial parameters at size  $2N$  come from the parameters obtained at size  $N$ , at the completion of its simulation. Therefore, the compute time for size  $2N$  in the transfer learning case is the sum of computation time for size  $N$  and computation time taken for size  $2N$  itself to meet the stopping criterion, after being initialised with the transferred parameters. The sequence starts with a system of size  $N = 4$  being randomly initialised with a cold-start.

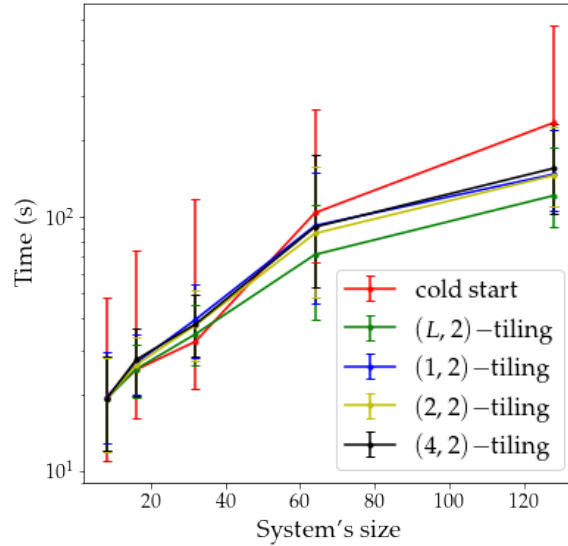
Figures 3.10 to 3.13 plot the convergence time for cold-start as well as transfer learning using all four tiling protocols for systems of sizes 8, 16, 32, 64 and 128. Each of these 4 figures shows results for the Ising  $J_1 - J_2$  model with  $(J_1 = -0.5, J_2 = -5.0)$ ,  $(J_1 = -0.5, J_2 = 5.0)$ ,  $(J_1 = 0.5, J_2 = -5.0)$ , and  $(J_1 = 0.5, J_2 = 5.0)$ , respectively. The vertical error bars show the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations, with shorter bars implying more consistency across the iterations for that system size.

It is evident that  $(L, 2) - \text{tiling}$  performs the best among the tiling protocols, and even

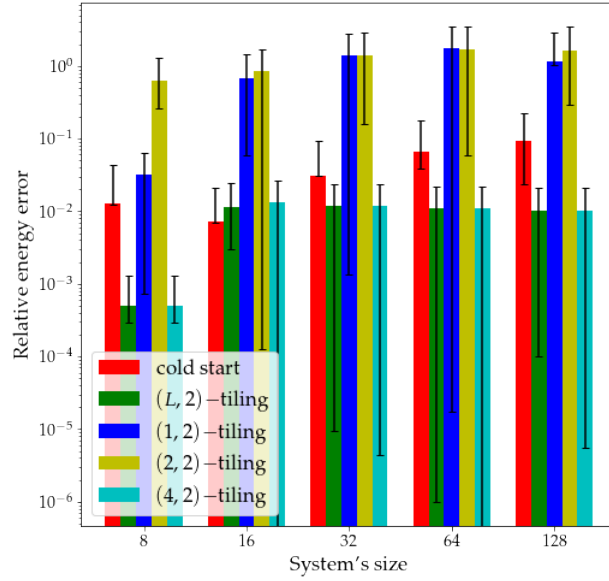




**Figure 3.4:** Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = 0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



**Figure 3.5:** Efficiency of cold-start and different tiling protocols when using the RBM architecture, with  $J_1 = 0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *ferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



**Figure 3.6:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

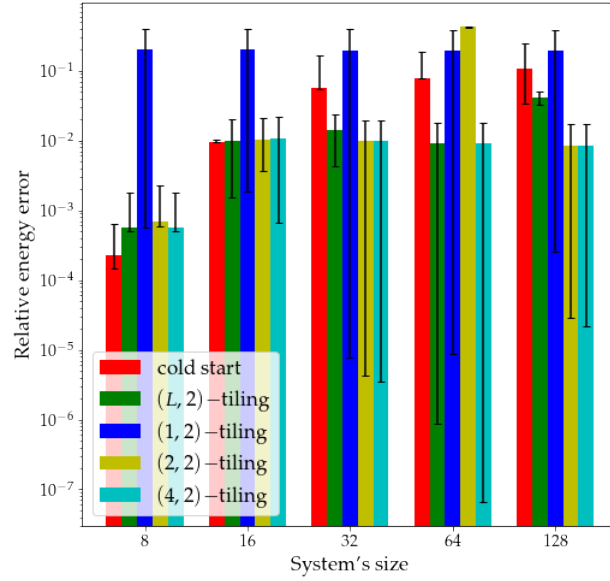
gains over cold-start for systems of larger size. Note that the different values of  $J_1$  and  $J_2$  correspond to the different phases and the  $(L, 2)$  - *tiling* protocol has least time to convergence across all of them. It can also be seen that the benefits of transfer learning in terms of efficiency get more pronounced with increase in system size.

Similar to the RBM, Figures 3.14 to 3.17 show the relative error  $\Delta E/E_0$  from the different system sizes— 8, 16, 32, 64, 128 for cold start as well as the  $(1, 2)$ ,  $(2, 2)$ ,  $(4, 2)$  and  $(L, 2)$  tiling protocols, when the stopping criterion is reached. Each of these 4 figures plot results for the Ising  $J_1 - J_2$  model with  $(J_1 = -0.5, J_2 = -5.0)$ ,  $(J_1 = -0.5, J_2 = 5.0)$ ,  $(J_1 = 0.5, J_2 = -5.0)$ , and  $(J_1 = 0.5, J_2 = 5.0)$ , respectively, with the vertical bars representing the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

It is evident that among the transfer learning protocols,  $(L, 2)$  has the lowest relative error for most scenarios. Note that interpreting the effectiveness of each of the tiling protocols in case of the MLP is trickier than the RBM case, owing to the presence of the hidden layer.

### 3.3 Conclusions

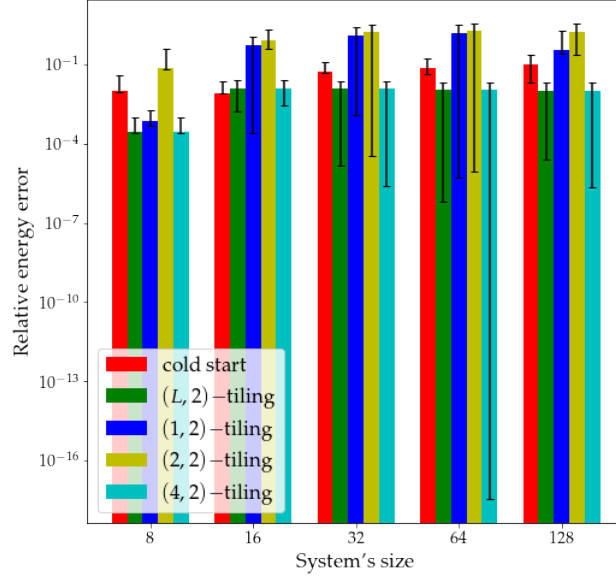
In this chapter, we looked at the various  $(k, p)$  - *tiling* protocols [102] and how they perform with regards to the one-dimensional Ising  $J_1 - J_2$  model. We evaluated them against yardsticks



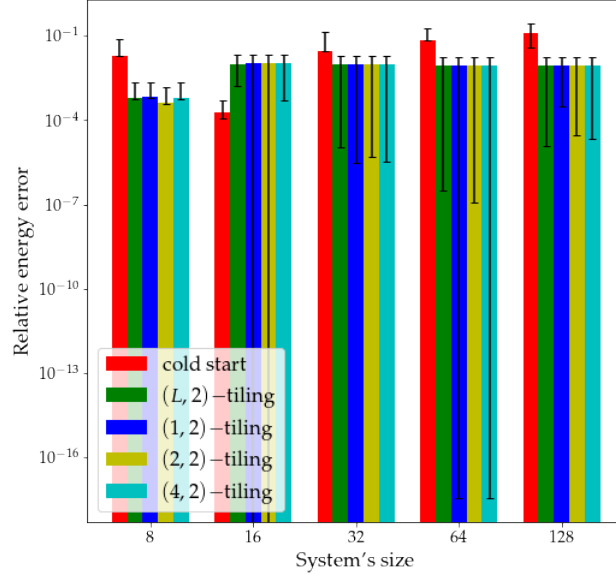
**Figure 3.7:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.

of efficiency and effectiveness in different phases of the model. The best performing transfer learning protocol is the  $(L, 2)$ -tiling protocol, which was effective across the different phases for all the system sizes we considered. Our empirical results demonstrate that employing transfer learning for the scalability of neural quantum states (when determining ground states) can be both (i) efficient: as it provides shorter convergence times, in comparison to cold-start runs, particularly for larger systems, and (ii) effective: as it reduces the chances of the optimization being stuck in a local minima by initialising from a converged base network. It must also be noted that the choice of tiling protocol is quite critical. For instance, as we saw in Sec. 3.2.1, both  $(1, 2)$  and  $(2, 2)$  tiling may favor ferromagnetic phase transfers, but they perform poorly in the next-nearest-neighbor antiferromagnetic phase. Given a configuration  $|\uparrow\uparrow\downarrow\downarrow\rangle$  in the next-nearest antiferromagnetic phase,  $(1, 2)$ -tiling will give bias towards  $|\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow\rangle$  while  $(2, 2)$ -tiling will give bias towards  $|\uparrow\uparrow\uparrow\uparrow\downarrow\downarrow\downarrow\downarrow\rangle$ , both of which imply weaker next-nearest neighbor antiferromagnetic correlations.

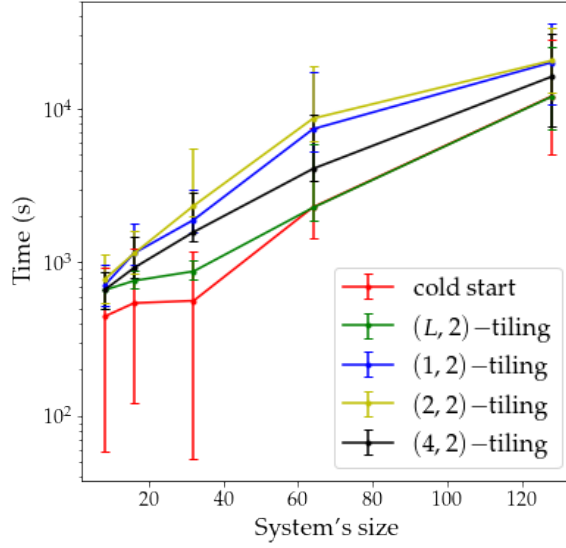
In summary, if the characteristic traits (i.e., the relative spin alignments) of the target ground state phase are not preserved in the transfer learning protocol, the transfer learning method becomes ineffective and inefficient. On the other hand, protocols that preserve such traits (such as  $(L, 2)$ -tiling, by repeating the trends in the entirety of the system during transfer, see schematic in Fig. 3.1(c)) are generally stable.



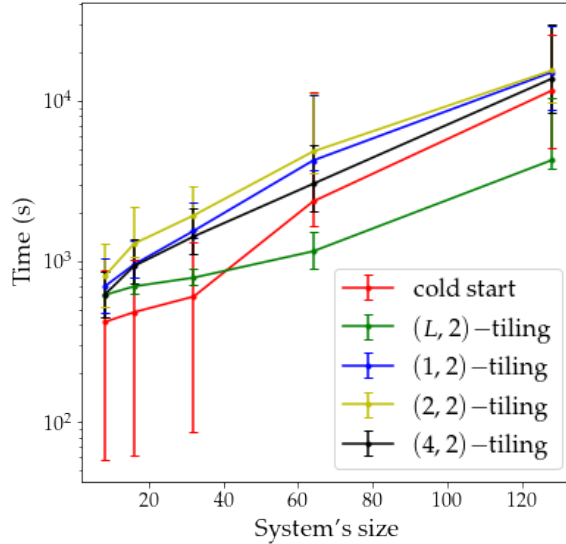
**Figure 3.8:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



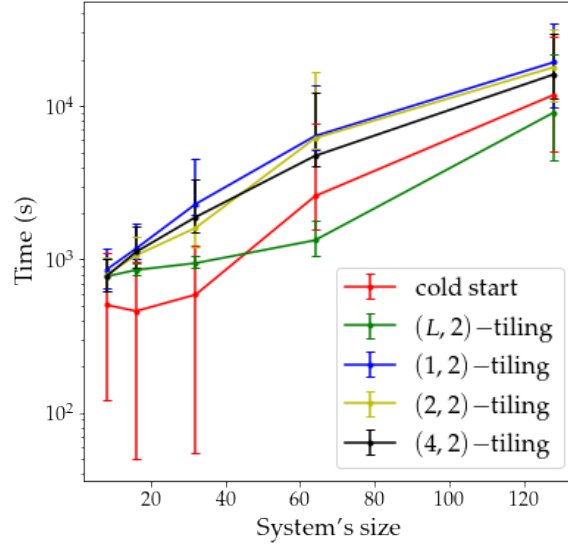
**Figure 3.9:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the RBM architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *ferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



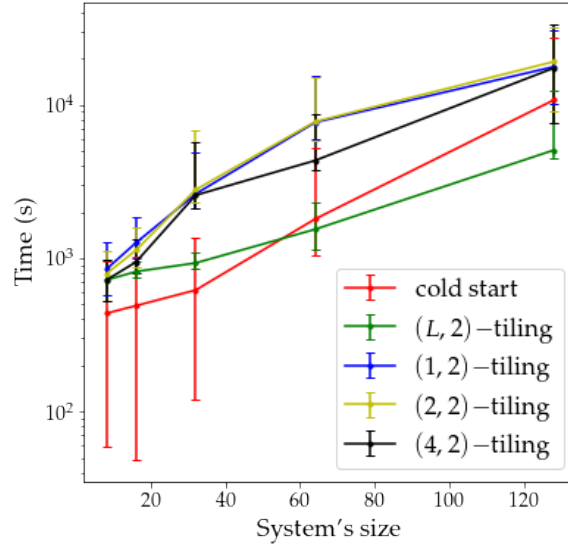
**Figure 3.10:** The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = -0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



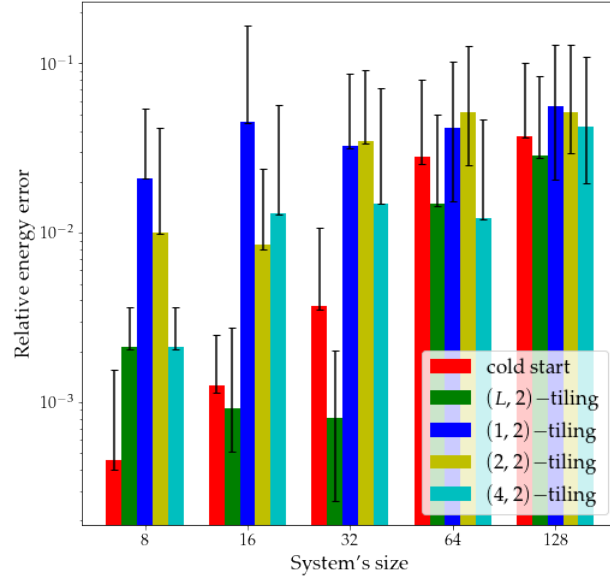
**Figure 3.11:** The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = -0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



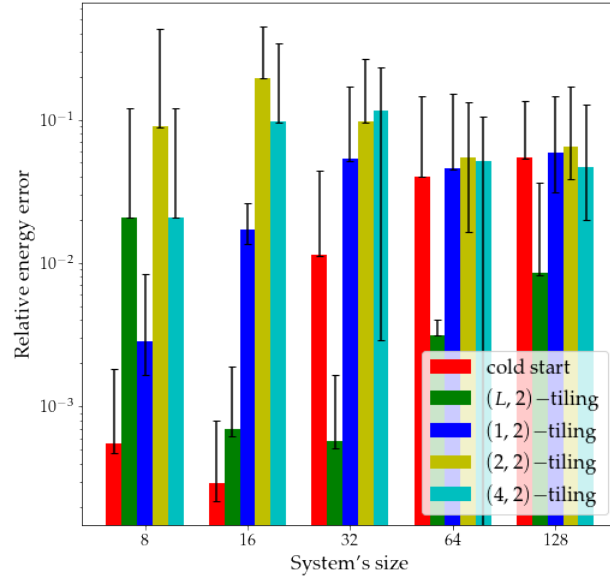
**Figure 3.12:** The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = 0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



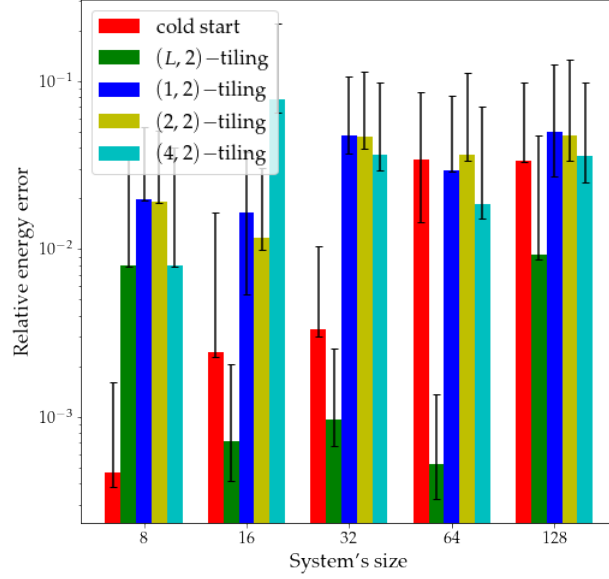
**Figure 3.13:** The efficiency of cold-start and different tiling protocols when using the MLP architecture, with  $J_1 = 0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *ferromagnetic phase*. Plot shows the time it took to meet the stopping criterion (in seconds) for different transfer learning protocols and cold-start, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



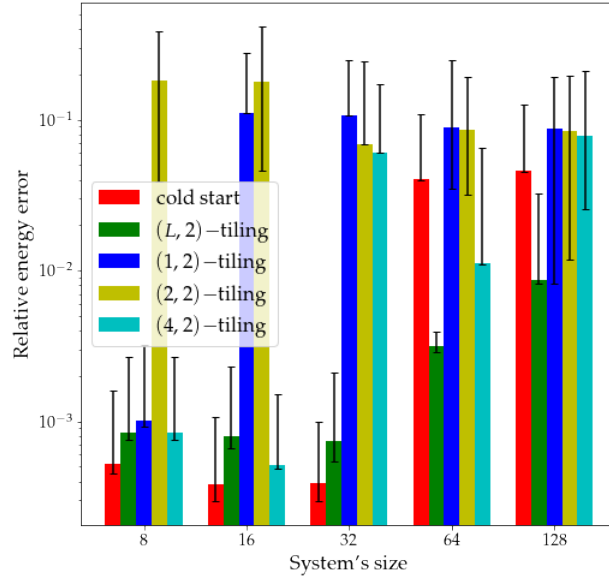
**Figure 3.14:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = -5.0, h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



**Figure 3.15:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = -0.5, J_2 = 5.0, h = 1.0$ , i.e., *nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



**Figure 3.16:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5$ ,  $J_2 = -5.0$ ,  $h = 1.0$ , i.e., *next-nearest-neighbor antiferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



**Figure 3.17:** Relative ground state energy errors  $\Delta E/E_0$  for different transfer learning protocols and cold-start (relative to accurate matrix product states values) using the MLP architecture, as a function of the number of spins  $N = \{8, 16, 32, 64, 128\}$ . Here  $J_1 = 0.5$ ,  $J_2 = 5.0$ ,  $h = 1.0$ , i.e., *ferromagnetic phase*. Error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 20 realisations.



# CHAPTER 4

## Excited States Energy levels for Ising model using NQS

In this chapter, we describe the methods we employed to determine the excited state energy levels. Note that the model we considered for these methods is the simple Ising model (Sec. 2.1.2), which only accounts for nearest-neighbor interactions. The following section introduces said methods with sections 4.1.1 and 4.1.2 going into specific details of each method. The performance evaluation for each of these methods is given in Sec. 4.2, followed by a summary of results and conclusion in Sec. 4.3.

### 4.1 Excited States Methods

In order to determine excited states, we use the fact that the various energy levels that a quantum system can be in (whether it is in the ground state or an excited state), are exactly dictated by the various eigenvalues of the Hamiltonian matrix for the system. We also know that minimizing the expectation of local energy helps us find the lowest eigenvalue, i.e., the ground state energy. Therefore, by modifying the Hamiltonian matrix such that the eigenvalues are rearranged, we can expect the minimization to converge to the eigenvalues for the first excited state. Similarly, we can repeat the process to find subsequent excited states.

As part of this work, we employed two methods that rearrange the eigenvalues of the system. The first method does so as a constraint while the second one constructs a different Hamiltonian. The former is similar to the existing work on determining excited states [62] which uses orthogonal constraints to rearrange the eigenvalues of one and two dimensional quantum harmonic oscillator systems. In the remaining text, we refer to this method as Method-I. The latter method constructs a new Hamiltonian from the existing one, such that the eigenvalues are shifted. We refer to this approach as Method-II from hereon. The objective function used in Method-II is the same as the one used in finding the ground state, i.e., that given in Eq. 3.1, except that the Hamiltonian matrix gets replaced by the modified matrix for every energy level.

In the following subsections, we go into the details of each of these methods.

### 4.1.1 Excited States using Method-I

As explained in the previous sections, minimizing the average energy of a quantum system (Eq. 2.13) gets us to the ground state. In order to find the first excited state, we use the ground state as a prior and minimize the following expression, similar to [62]:

$$E_{1_\Psi} = \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} + \lambda \frac{\langle \Psi | \Psi_0 \rangle \langle \Psi_0 | \Psi \rangle}{\langle \Psi | \Psi \rangle \langle \Psi_0 | \Psi_0 \rangle} \quad (4.1)$$

where  $\Psi$  is the neural network representation of the state of the quantum many-body system,  $\Psi_0$  is the ground state wave function and  $\lambda$  is a hyperparameter that controls the effect of the penalty term. In general, to find the  $N^{th}$  excited state, we enforce orthogonality with all previous excited states as indicated by the following expression:

$$E_{N_\Psi} = \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} + \lambda \sum_{i=0}^{N-1} \left( \frac{\langle \Psi | \Psi_i \rangle \langle \Psi_i | \Psi \rangle}{\langle \Psi | \Psi \rangle \langle \Psi_i | \Psi_i \rangle} \right) \quad (4.2)$$

Considering the first excited state for now, from Eq. 2.12 and Eq. 2.14, Eq. 4.1 becomes

$$E_{1_\Psi} = \sum_{\mathbf{x}} p(x) E_{loc}(\mathbf{x}) + \lambda \frac{\langle \Psi | \Psi_0 \rangle \langle \Psi_0 | \Psi \rangle}{\langle \Psi | \Psi \rangle \langle \Psi_0 | \Psi_0 \rangle} \quad (4.3)$$

Now, for the second term,

$$\lambda \frac{\langle \Psi | \Psi_0 \rangle \langle \Psi_0 | \Psi \rangle}{\langle \Psi | \Psi \rangle \langle \Psi_0 | \Psi_0 \rangle} = \lambda \left( \frac{\sum_x \langle \psi | x \rangle \langle x | \psi_0 \rangle}{\sum_k |\psi(k)|^2} \right) \cdot \left( \frac{\sum_n \langle \psi_0 | n \rangle \langle n | \psi \rangle}{\sum_j |\psi_0(j)|^2} \right) \quad (4.4)$$

$$= \lambda \left( \frac{\sum_x \psi^*(x) \psi_0(x)}{\sum_k |\psi(k)|^2} \right) \cdot \left( \frac{\sum_n \psi_0^*(n) \psi(n)}{\sum_j |\psi_0(j)|^2} \right) \quad (4.5)$$

where  $\psi$  is the neural network representation of the system state, while  $\psi_0$  represents the wave function for the ground state;  $x$  represents the samples for the excited state, obtained via Monte Carlo sampling, while  $n$  represents the samples from the converged ground state (also obtained via Monte Carlo sampling during prior training).

Multiplying the numerators of both the terms by  $\frac{\psi(x)}{\psi(x)}$  and  $\frac{\psi_0(n)}{\psi_0(n)}$ , respectively

$$= \lambda \left( \frac{\sum_x |\psi(x)|^2 \frac{\psi_0(x)}{\psi(x)}}{\sum_k |\psi(k)|^2} \right) \cdot \left( \frac{\sum_n |\psi_0(n)|^2 \frac{\psi(n)}{\psi_0(n)}}{\sum_j |\psi_0(j)|^2} \right) \quad (4.6)$$

$$= \lambda \sum_{x,n} P_\psi(x) P_{\psi_0}(n) E_{\psi_0 \psi}(x) E_{\psi \psi_0}(n) \quad (4.7)$$

where

$$P_\psi(x) = \frac{|\psi(x)|^2}{\sum_k |\psi(k)|^2} \text{ and } P_{\psi_0}(n) = \frac{|\psi_0(n)|^2}{\sum_j |\psi_0(j)|^2}, \text{ representing probability distributions;}$$

$$\text{and } E_{\psi_0\psi}(x) = \frac{\psi_0(x)}{\psi(x)} \text{ and } E_{\psi\psi_0}(n) = \frac{\psi(n)}{\psi_0(n)}$$

So Eq. 4.7 can be rewritten as

$$\lambda \frac{\langle \Psi | \Psi_0 \rangle \langle \Psi_0 | \Psi \rangle}{\langle \Psi | \Psi \rangle \langle \Psi_0 | \Psi_0 \rangle} = \lambda \sum_x P_\psi(x) O_{loc}(x) \quad (4.8)$$

$$\text{where } O_{loc}(x) = \sum_n P_{\psi_0}(n) E_{\psi_0\psi}(x) E_{\psi\psi_0}(n)$$

From equations 4.3 and 4.8,

$$E_{1_\Psi} = \sum_{\mathbf{x}} p(\mathbf{x}) E_{loc}(\mathbf{x}) + \lambda \sum_{\mathbf{x}} P_\psi(\mathbf{x}) O_{loc}(\mathbf{x}) \quad (4.9)$$

As  $P_\psi(\mathbf{x})$  is also a probability distribution over the random variable  $\mathbf{x}$ , it is the same as  $p(\mathbf{x})$ . Therefore, for the first excited state,

$$\begin{aligned} E_{1_\Psi} &= \sum_{\mathbf{x}} p(\mathbf{x}) E_{loc}(\mathbf{x}) + \lambda \sum_{\mathbf{x}} p(\mathbf{x}) O_{loc}(\mathbf{x}) \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) \left( E_{loc}(\mathbf{x}) + \lambda O_{loc}(\mathbf{x}) \right) \end{aligned} \quad (4.10)$$

and in general,

$$E_\Psi = \sum_{\mathbf{x}} p(\mathbf{x}) \left( E_{loc}(\mathbf{x}) + \lambda \sum_{i=0}^{N-1} O_{loc}(\mathbf{x}) \right) \quad (4.11)$$

$$\text{where } O_{loc}(\mathbf{x}) = \sum_n P_{\psi_i}(n) E_{\psi_i\psi}(\mathbf{x}) E_{\psi\psi_i}(n)$$

$n$  represents the samples from the  $i^{th}$  prior state

$N$  is the highest excited state level to be determined

Thus compared to the ground state energy, the excited state energy has an additional orthogonal penalty term.

### Gradient of the Objective Function

Considering the objective function for the first excited state, we have,

$$\mathcal{L}_{exc}(\theta) = E_{1_\Psi} = \sum_{\mathbf{x}} p(\mathbf{x}) E_{loc}(\mathbf{x}) + \lambda \sum_{\mathbf{x}} p(\mathbf{x}) O_{loc}(\mathbf{x}) \quad (4.12)$$

Taking the gradient of  $E_{1\psi}$  with respect to the weights  $\theta$ , note that the gradient of the first term is the same as that in Section 3.2, detailed in [55]

Now for the second term,

$$\begin{aligned}
 & \lambda \frac{\partial}{\partial \theta} \sum_{\mathbf{x}} p(\mathbf{x}, \theta) O_{loc}(\mathbf{x}, \theta) \\
 &= \lambda \frac{\partial}{\partial \theta} \left( \sum_{\mathbf{x}, \mathbf{n}} P_{\psi}(\mathbf{x}, \theta) P_{\psi_0}(\mathbf{n}) E_{\psi_0 \psi}(\mathbf{x}, \theta) E_{\psi \psi_0}(\mathbf{n}, \theta) \right) \\
 &= \lambda \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) \left[ \left( \frac{\partial}{\partial \theta} P_{\psi}(\mathbf{x}, \theta) \right) E_{\psi_0 \psi}(\mathbf{x}, \theta) E_{\psi \psi_0}(\mathbf{n}, \theta) + \left( \frac{\partial}{\partial \theta} E_{\psi_0 \psi}(\mathbf{x}, \theta) \right) P_{\psi}(\mathbf{x}, \theta) E_{\psi \psi_0}(\mathbf{n}, \theta) \right. \\
 &\quad \left. + \left( \frac{\partial}{\partial \theta} E_{\psi \psi_0}(\mathbf{n}, \theta) \right) P_{\psi}(\mathbf{x}, \theta) E_{\psi_0 \psi}(\mathbf{x}, \theta) \right] \tag{4.13}
 \end{aligned}$$

For  $\frac{\partial}{\partial \theta} P_{\psi}(\mathbf{x}, \theta)$ , we have

$$\begin{aligned}
 \frac{\partial}{\partial \theta} P_{\psi}(\mathbf{x}, \theta) &= \frac{\partial}{\partial \theta} \frac{|\psi(x)|^2}{\sum_k |\psi(k)|^2} = \frac{\partial}{\partial \theta} \frac{|\psi(x)|^2}{Z(\theta)} \text{ where } Z = \sum_k |\psi(k)|^2 \\
 &= \frac{\partial}{\partial \theta} \frac{\psi^*(\mathbf{x}, \theta) \psi(\mathbf{x}, \theta)}{Z(\theta)} \\
 &= \frac{\psi(\mathbf{x}, \theta)}{Z(\theta)} \frac{\partial \psi^*(\mathbf{x}, \theta)}{\partial \theta} + \frac{\psi^*(\mathbf{x}, \theta)}{Z(\theta)} \frac{\partial \psi(\mathbf{x}, \theta)}{\partial \theta} - \frac{\psi^*(\mathbf{x}, \theta) \psi(\mathbf{x}, \theta)}{Z^2(\theta)} \frac{\partial Z(\theta)}{\partial \theta} \tag{4.14}
 \end{aligned}$$

where

$$\begin{aligned}
 \frac{\partial Z(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \left( \sum_{\mathbf{k}} \psi^*(\mathbf{k}, \theta) \psi(\mathbf{k}, \theta) \right) \\
 &= \sum_{\mathbf{k}} \frac{\partial \psi^*(\mathbf{k}, \theta)}{\partial \theta} \psi(\mathbf{k}, \theta) + \psi^*(\mathbf{k}, \theta) \frac{\partial \psi(\mathbf{k}, \theta)}{\partial \theta} \\
 &= \sum_{\mathbf{k}} \frac{1}{\psi^*(\mathbf{k}, \theta)} \frac{\partial \psi^*(\mathbf{k}, \theta)}{\partial \theta} |\psi(\mathbf{k}, \theta)|^2 + \frac{1}{\psi(\mathbf{k}, \theta)} \frac{\partial \psi(\mathbf{k}, \theta)}{\partial \theta} |\psi(\mathbf{k}, \theta)|^2 \tag{4.15}
 \end{aligned}$$

Therefore,

$$\begin{aligned}
 \frac{\partial}{\partial \theta} P_{\psi}(\mathbf{x}, \theta) &= \frac{|\psi(\mathbf{x}, \theta)|^2}{Z(\theta)} \frac{1}{\psi^*(\mathbf{x}, \theta)} \frac{\partial \psi^*(\mathbf{x}, \theta)}{\partial \theta} + \frac{|\psi(\mathbf{x}, \theta)|^2}{Z(\theta)} \frac{1}{\psi(\mathbf{x}, \theta)} \frac{\partial \psi(\mathbf{x}, \theta)}{\partial \theta} \\
 &\quad - \frac{|\psi(\mathbf{x}, \theta)|^2}{Z(\theta)} \sum_{\mathbf{k}} \frac{1}{\psi^*(\mathbf{k}, \theta)} \frac{\partial \psi^*(\mathbf{k}, \theta)}{\partial \theta} \frac{|\psi(\mathbf{k}, \theta)|^2}{Z(\theta)} \\
 &\quad + \frac{1}{\psi(\mathbf{k}, \theta)} \frac{\partial \psi(\mathbf{k}, \theta)}{\partial \theta} \frac{|\psi(\mathbf{k}, \theta)|^2}{Z(\theta)} \tag{4.16}
 \end{aligned}$$

Now, let

$$D_{\theta}(\mathbf{x}, \theta) = \frac{1}{\psi(\mathbf{x}, \theta)} \frac{\partial(\mathbf{x}, \theta)}{\partial \theta}$$

Rewriting the gradient of  $\frac{\partial}{\partial \theta} P_\psi(\mathbf{x}, \theta)$ ,

$$\begin{aligned}
 \frac{\partial}{\partial \theta} P_\psi(\mathbf{x}, \theta) &= P_\psi(\mathbf{x}, \theta) D_\theta^*(\mathbf{x}, \theta) + P_\psi(\mathbf{x}, \theta) D_\theta(\mathbf{x}, \theta) \\
 &\quad - P_\psi(\mathbf{x}, \theta) \sum_{\mathbf{k}} P_\psi(\mathbf{k}, \theta) D_\theta^*(\mathbf{k}, \theta) + P_\psi(\mathbf{k}, \theta) D_\theta(\mathbf{k}, \theta) \\
 &= P_\psi(\mathbf{x}, \theta) D_\theta^*(\mathbf{x}, \theta) + P_\psi(\mathbf{x}, \theta) D_\theta(\mathbf{x}, \theta) - P_\psi(\mathbf{x}, \theta) (\mathbb{E}[D_\theta^*] + \mathbb{E}[D_\theta]) \quad (4.17)
 \end{aligned}$$

where the expectation is calculated over the distribution  $P_\psi(\mathbf{x}, \theta)$

For  $\frac{\partial}{\partial \theta} E_{\psi_0 \psi}(\mathbf{x}, \theta)$ , we have

$$\begin{aligned}
 \frac{\partial}{\partial \theta} E_{\psi_0 \psi}(\mathbf{x}, \theta) &= \frac{\partial}{\partial \theta} \frac{\psi_0(\mathbf{x})}{\psi(\mathbf{x}, \theta)} \\
 &= -\psi_0(\mathbf{x}) \frac{1}{\psi^2(\mathbf{x}, \theta)} \frac{\partial \psi(\mathbf{x}, \theta)}{\partial \theta} = -\frac{\psi_0(\mathbf{x})}{\psi(\mathbf{x}, \theta)} \frac{1}{\psi(\mathbf{x}, \theta)} \frac{\partial \psi(\mathbf{x}, \theta)}{\partial \theta} \\
 &= -E_{\psi_0 \psi}(\mathbf{x}, \theta) D_\theta(\mathbf{x}, \theta) \quad (4.18)
 \end{aligned}$$

And for  $\frac{\partial}{\partial \theta} E_{\psi \psi_0}(\mathbf{n}, \theta)$ , we have

$$\begin{aligned}
 \frac{\partial}{\partial \theta} E_{\psi \psi_0}(\mathbf{n}, \theta) &= \frac{\partial}{\partial \theta} \frac{\psi(\mathbf{n}, \theta)}{\psi_0(\mathbf{n})} \\
 &= \frac{1}{\psi_0(\mathbf{n})} \frac{\partial \psi(\mathbf{n}, \theta)}{\partial \theta} = \frac{\psi(\mathbf{n}, \theta)}{\psi_0(\mathbf{n})} \frac{1}{\psi(\mathbf{n}, \theta)} \frac{\partial \psi(\mathbf{n}, \theta)}{\partial \theta} \\
 &= E_{\psi \psi_0}(\mathbf{n}, \theta) D_\theta(\mathbf{n}, \theta) \quad (4.19)
 \end{aligned}$$

Now, using the above values in Eq. 4.13,

$$\begin{aligned}
 & \lambda \frac{\partial}{\partial \theta} \sum_{\mathbf{x}} p(\mathbf{x}, \boldsymbol{\theta}) O_{loc}(\mathbf{x}, \boldsymbol{\theta}) \\
 &= \lambda \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) \left[ \left( \frac{\partial}{\partial \theta} P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) \right) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) + \left( \frac{\partial}{\partial \theta} E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \right) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) \right. \\
 & \quad \left. + \left( \frac{\partial}{\partial \theta} E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) \right) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \right] \\
 &= \lambda \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) \left[ \left( E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) D_{\theta}^*(\mathbf{x}, \boldsymbol{\theta}) + E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) D_{\theta}(\mathbf{x}, \boldsymbol{\theta}) \right. \right. \\
 & \quad \left. \left. - E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) (\mathbb{E}[D_{\theta}^*] + \mathbb{E}[D_{\theta}]) \right) + \left( - E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) D_{\theta}(\mathbf{x}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) \right) \right. \\
 & \quad \left. + \left( E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) D_{\theta}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \right) \right] \\
 &= \lambda \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) \left[ E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) \left( D_{\theta}^*(\mathbf{x}, \boldsymbol{\theta}) + D_{\theta}(\mathbf{x}, \boldsymbol{\theta}) - (\mathbb{E}[D_{\theta}^*] + \mathbb{E}[D_{\theta}]) \right) \right. \\
 & \quad \left. - D_{\theta}(\mathbf{x}, \boldsymbol{\theta}) + D_{\theta}(\mathbf{n}, \boldsymbol{\theta}) \right] \\
 &= \lambda \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) \left[ E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) \left( D_{\theta}^*(\mathbf{x}, \boldsymbol{\theta}) - \mathbb{E}[D_{\theta}^*] - \mathbb{E}[D_{\theta}] + D_{\theta}(\mathbf{n}, \boldsymbol{\theta}) \right) \right] \\
 &= \lambda \left[ \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) D_{\theta}^*(\mathbf{x}, \boldsymbol{\theta}) \right. \\
 & \quad - \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \mathbb{E}[D_{\theta}^*] \\
 & \quad - \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \mathbb{E}[D_{\theta}] \\
 & \quad \left. + \sum_{\mathbf{x}, \mathbf{n}} P_{\psi_0}(\mathbf{n}) E_{\psi \psi_0}(\mathbf{n}, \boldsymbol{\theta}) D_{\theta}(\mathbf{n}, \boldsymbol{\theta}) P_{\psi}(\mathbf{x}, \boldsymbol{\theta}) E_{\psi_0 \psi}(\mathbf{x}, \boldsymbol{\theta}) \right]
 \end{aligned}$$

Rewriting the above, in the form of Expectation over respective distributions,

$$\begin{aligned}
 &= \lambda \left[ \mathbb{E}_{\psi_0}[E_{\psi \psi_0}] \mathbb{E}[E_{\psi_0 \psi} D_{\theta}^*] - \mathbb{E}_{\psi_0}[E_{\psi \psi_0}] \mathbb{E}[E_{\psi_0 \psi}] \mathbb{E}[D_{\theta}^*] - \mathbb{E}_{\psi_0}[E_{\psi \psi_0}] \mathbb{E}[E_{\psi_0 \psi}] \mathbb{E}[D_{\theta}] \right. \\
 & \quad \left. + \mathbb{E}_{\psi_0}[E_{\psi \psi_0} D_{\theta}] \mathbb{E}[E_{\psi_0 \psi}] \right] \\
 &= \lambda \left[ \mathbb{E}_{\psi_0}[E_{\psi \psi_0}] \mathbb{E}[E_{\psi_0 \psi} D_{\theta}^*] - \mathbb{E}_{\psi_0}[E_{\psi \psi_0}] \mathbb{E}[E_{\psi_0 \psi}] \left( \mathbb{E}[D_{\theta}^*] + \mathbb{E}[D_{\theta}] \right) + \mathbb{E}_{\psi_0}[E_{\psi \psi_0} D_{\theta}] \mathbb{E}[E_{\psi_0 \psi}] \right]
 \end{aligned} \tag{4.20}$$

Finally, from Eq. 3.2 and Eq. 4.20,

$$\frac{\partial \mathcal{L}}{\partial \theta} = 2 \operatorname{Re}(\mathbb{E}[E_{loc} D_{\theta}^*] - \mathbb{E}[E_{loc}] \mathbb{E}[D_{\theta}^*]) + \lambda \left[ \mathbb{E}_{\psi_0}[E_{\psi\psi_0}] \mathbb{E}[E_{\psi_0\psi} D_{\theta}^*] - \mathbb{E}_{\psi_0}[E_{\psi\psi_0}] \mathbb{E}[E_{\psi_0\psi}] \left( \mathbb{E}[D_{\theta}^*] + \mathbb{E}[D_{\theta}] \right) + \mathbb{E}_{\psi_0}[E_{\psi\psi_0} D_{\theta}] \mathbb{E}[E_{\psi_0\psi}] \right]$$

where  $\mathbb{E}_{\psi_0}$  denotes expectation over the distribution  $P_{\psi_0}$  and  $\mathbb{E}$  denotes expectation over the distribution  $P_{\psi}$

So in general, following from Eq. 4.2, the gradient would be:

$$2 \operatorname{Re}(\mathbb{E}[E_{loc} D_{\theta}^*] - \mathbb{E}[E_{loc}] \mathbb{E}[D_{\theta}^*]) + \lambda \left[ \sum_{i=0}^{N-1} \mathbb{E}_{\psi_i}[E_{\psi\psi_i}] \mathbb{E}[E_{\psi_i\psi} D_{\theta}^*] - \sum_{i=0}^{N-1} \mathbb{E}_{\psi_i}[E_{\psi\psi_i}] \mathbb{E}[E_{\psi_i\psi}] \left( \mathbb{E}[D_{\theta}^*] + \mathbb{E}[D_{\theta}] \right) + \sum_{i=0}^{N-1} \mathbb{E}_{\psi_i}[E_{\psi\psi_i} D_{\theta}] \mathbb{E}[E_{\psi_i\psi}] \right] \quad (4.21)$$

With that we have described the objective function and the gradient objective function for determining excited states using Method-I. We next look at the details of Method-II.

### 4.1.2 Excited States using Method-II

In this section, we present an approach for finding the excited states by shifting the eigenvalues of the Hamiltonian to zero, for every excited state we wish to determine. Formally, the approach is as follows.

First, we determine the ground state energy, following the approach described in the previous chapter. Let  $H_0$  be the Hamiltonian corresponding to the system at this state. Next, for the first excited state, we modify the Hamiltonian as in Eq. 4.22, where  $\psi_0$  and  $e_0$  are the ground state wave function and energy, respectively, when the model (NNQS) converges.

$$H_1 = H_0 - \left( e_0 \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right) \quad (4.22)$$

$$H_2 = H_1 - \left( e_1 \frac{|\psi_1\rangle \langle \psi_1|}{\langle \psi_1 | \psi_1 \rangle} \right) \quad (4.23)$$

Then we use this Hamiltonian and minimize the expectation of local energy using the same objective function as in Eq. 3.1. Once training converges, we modify the Hamiltonian again as in Eq. 4.23, to subsequently determine the second excited state. Similarly, we can continue in this manner to find subsequent excited states.

Starting with,

$$\langle \psi | H | \psi \rangle = \langle \psi | E | \psi \rangle \implies H | \psi \rangle \langle \psi | = E | \psi \rangle \langle \psi | \implies H = E \frac{|\psi\rangle \langle \psi|}{\langle \psi | \psi \rangle} \quad (4.24)$$

$$\begin{aligned} \text{As } \mathbf{E} &= \sum_{i=0}^{N-1} e_i, \text{ where } N \text{ is system size and } e_i \text{ are the eigenvalues of } \mathbf{E}, \\ \implies H &= \sum_{i=0}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) \end{aligned} \quad (4.25)$$

Using this in Eq. 4.22,

$$H_1 = \sum_{i=0}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) - e_0 \left( \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right) \quad (4.26)$$

$$\implies H_1 = \sum_{i=1}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) + \cancel{e_0 \left( \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right)} - \cancel{e_0 \left( \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right)} \quad (4.27)$$

Similarly,

$$H_2 = H_1 - e_1 \left( \frac{|\psi_1\rangle \langle \psi_1|}{\langle \psi_1 | \psi_1 \rangle} \right) \quad (4.28)$$

Using Eq. 4.27 and taking  $e_1$  out of the summation,

$$\begin{aligned} H_2 &= \sum_{i=2}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) + \cancel{e_0 \left( \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right)} - \cancel{e_0 \left( \frac{|\psi_0\rangle \langle \psi_0|}{\langle \psi_0 | \psi_0 \rangle} \right)} \\ &\quad + \cancel{e_1 \left( \frac{|\psi_1\rangle \langle \psi_1|}{\langle \psi_1 | \psi_1 \rangle} \right)} - \cancel{e_1 \left( \frac{|\psi_1\rangle \langle \psi_1|}{\langle \psi_1 | \psi_1 \rangle} \right)} \end{aligned} \quad (4.29)$$

That is,

$$H_k = \sum_{i=k}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) + \sum_{j=0}^{k-1} \cancel{e_j \left( \frac{|\psi_j\rangle \langle \psi_j|}{\langle \psi_j | \psi_j \rangle} \right)} - \cancel{e_j \left( \frac{|\psi_j\rangle \langle \psi_j|}{\langle \psi_j | \psi_j \rangle} \right)} \quad (4.30)$$

So at every excited state, the Hamiltonian gets modified through a shifting of the eigenvalues (via the second term) to zero. The minimization is thus able to find the new ground state, i.e.,  $\psi_k$ , corresponding to the lowest eigenvalue in  $H_k$  from Eq. 4.31. That is,

$$H_k = \sum_{i=k}^{N-1} e_i \left( \frac{|\psi_i\rangle \langle \psi_i|}{\langle \psi_i | \psi_i \rangle} \right) \quad (4.31)$$

Note that in Eq. 4.22, the  $e_0$  could be replaced by any  $\kappa > e_1 - e_0$ . By using  $e_0$  the second term reduces to zero, but with such  $\kappa$  the eigenvalues would be raised, similar to Method-I described earlier. It should also be pointed out that this approach as it is, would not work if the desired energy level is positive. For the models we considered, we observe that



positive energy values show up in much higher excited states (in the 8<sup>th</sup> excited state for  $N = 4, J/h = 0.5$ , for instance). One possible way to handle such eigenvalues would be to use  $-H$  instead of  $H$  when determining such energy levels.

## 4.2 Performance Evaluation

In order to evaluate each of these methods in finding low-lying excited states for the one-dimensional simple Ising model, we implemented them for the MLP based NQS architecture, in the unsupervised learning setting.

To analyse the performance of the two methods over various phases, we fixed  $h = 1$  and varied  $J$  to take the values  $\{-1.0, -0.5, 0.5, 1.0\}$ . For the experiments in this chapter, we considered systems with size  $N = \{4, 8\}$  spins. The reported results are an average over 10 realisations of the same experiment as our calculations involved random components.

We evaluate the excited states exploration in terms of effectiveness, i.e., how accurately the system is able to approximate the excited state energy. As in the case of finding the ground state, the effectiveness is reported as the mean relative error between the predicted and true energy. The true energy values for these experiments were obtained from Exact Diagonalisation which is applicable only to systems with  $N < 20$ . Additionally, once the training converges, we measure its overlap with all prior states and verify that it indeed is orthogonal to them, implying a distinct quantum state. We also compare the two methods in terms of their efficiency, i.e., the time it took to get to the stopping criterion.

Similar to the ground state experiments, we use an iterative approach. At every iteration, we draw  $10^4$  samples to evaluate the energy  $\langle H \rangle_\Psi$  and its gradients using the parameters of the MLP. Note that in the case of Method-I, the calculation of energy includes the additional orthogonal penalty term (similarly for the calculation of the gradient). Next, the parameters of the MLP are updated using a gradient descent algorithm, specifically the **RMSProp** optimizer for its adaptive learning strategy. The initial learning rate was set to 0.001. For both the methods, samples were drawn using the Metropolis algorithm (specifically the **MetropolisAll** implementation, as the Hamiltonian matrix is not sparse anymore). As the eigenstates for excited states include complex numbers, we used a complex MLP with **tanh** activation function to accurately represent them. This is in contrast to the ground state, where using real and positive MLP (and RBM) sufficed. Through our preliminary runs for determining excited states, we found that having one hidden layer with thrice as many hidden nodes as the number of visible nodes performed well. Also, some parameters for the complex MLP had to be set randomly and we sampled those from the Normal distribution  $\mathcal{N}(0, 0.001)$ , i.e., with zero mean and standard deviation 0.001.

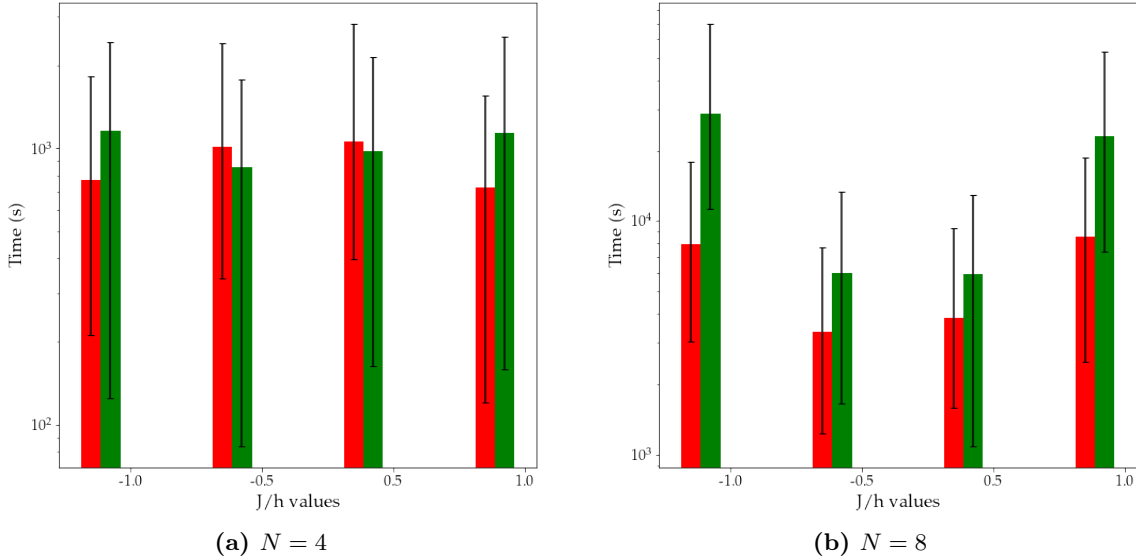
Similar to the ground state exploration, a dynamic stopping criterion was used for these experiments as well. For Method-I experiments, we set the threshold for the standard

deviation of local energy as well as the total energy to be 0.005 each. Additionally, we set the threshold for the total overlap (i.e., the dot product of the first excited state with the ground state) to be 0.001 to keep it close to zero. For Method-II, the stopping criterion considered only the standard deviation of local energy, which was set to 0.05. To account for cases when the stopping threshold was never met, the number of training epochs were capped at 20000.

All experiments in this chapter were run on an NVIDIA DGX-1 server equipped 5120 CUDA cores, and 16GB memory on the infrastructure of Singapore National Supercomputing Centre [1].

#### 4.2.1 Efficiency

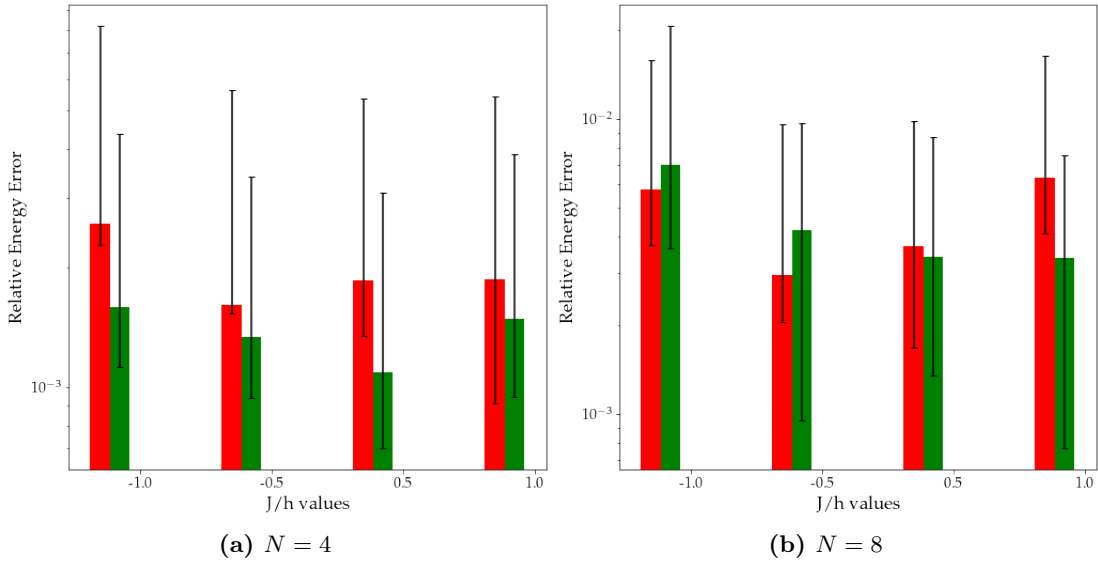
We first consider the efficiency of the two methods, for the one-dimensional Ising model, with  $J/h$  taking the values  $\{-1.0, -0.5, 0.5, 1.0\}$ . Fig. 4.1 shows the time to convergence for the two methods, with number of spins  $N = 4$  in Fig. 4.1a and  $N = 8$  in Fig. 4.1b. The values reported are from the epoch when the stopping criterion is reached and the error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained over 10 realisations. We observe that while the two methods can be considered comparably efficient for  $N = 4$ , Method-I becomes more efficient when  $N = 8$ , particularly for higher values of  $|J|/h$ .



**Figure 4.1:** Time to convergence for Method-I (shown in red) and Method-II (shown in green) across the different  $J/h$  values, with 4 and 8 spins, respectively. The error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 10 realisations.

### 4.2.2 Effectiveness

As a measure of the effectiveness of the two methods for the one-dimensional Ising model, Fig. 4.2 plots the relative energy error,  $\Delta E/E_1$  for the first excited state energy, across various values for  $J/h$ ,  $\{-1.0, -0.5, 0.5, 1.0\}$ , when  $N = 4$  in Fig. 4.2a and  $N = 8$  in Fig. 4.2b. The values reported are from the epoch when the stopping criterion is reached and the error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 10 realisations. We find both methods to be equally effective in finding the first excited state.



**Figure 4.2:** Relative first excited state energy error  $\Delta E/E_1$  for different values of  $J/h$  using Method-I (shown in red) and Method-II (shown in green), with 4 and 8 spins, respectively. The error bars indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, obtained from 10 realisations.

In addition to  $\Delta E/E_1$  as a measure of effectiveness, we also report the overlap between the first excited state determined using these methods and the ground state (obtained via cold-start runs as in Chapter 3). The overlap between two quantum states  $|\Omega\rangle$  and  $|\Phi\rangle$  is calculated as  $\frac{\langle\Phi|\Omega\rangle\langle\Omega|\Phi\rangle}{\langle\Omega|\Omega\rangle\langle\Phi|\Phi\rangle}$ , where  $\Omega$  and  $\Phi$  are the wave functions representing the two states. This overlap must evaluate to 0, if the two states are orthogonal to each other. As the excited states thus determined are quantum states distinct from the ground energy state, we expect the overlap values to at least be nearly 0, if not exactly.

Figures 4.3 and 4.4 tabulate the mean overlap values between the excited states determined by Method-I and Method-II, respectively. The reported values are calculated for each of the two system sizes, i.e., for  $N = 4$  and  $N = 8$ , and across the different values of  $J/h$ . We observe that the overlap is indeed close to 0, implying orthogonality of the two states, for all scenarios.

Size	$J/h$	Mean Overlap
4	-1.0	0.00031
	-0.5	0.00067
	0.5	0.00072
	1.0	0.00071

(a)  $N = 4$

Size	$J/h$	Mean Overlap
8	-1.0	0.00059
	-0.5	0.00049
	0.5	0.00079
	1.0	0.00069

(b)  $N = 8$

**Figure 4.3:** Tables listing the Mean Overlap values for the overlap between the ground state, and the first excited state obtained using Method-I, averaged over 10 iterations, for systems with 4 and 8 spins, respectively

Size	$J/h$	Mean Overlap
4	-1.0	0.00041
	-0.5	0.00075
	0.5	0.00089
	1.0	0.00080

(a)  $N = 4$

Size	$J/h$	Mean Overlap
8	-1.0	0.000102
	-0.5	0.000083
	0.5	0.000084
	1.0	0.000078

(b)  $N = 8$

**Figure 4.4:** Tables listing the Mean Overlap values for the overlap between the ground state, and the first excited state obtained using Method-II, averaged over 10 iterations, for systems with 4 and 8 spins, respectively

### 4.3 Conclusions

In this chapter, we looked at two methods to find low-lying excited states for the one-dimensional Ising model. We evaluated them for effectiveness across several values of  $J/h$  and observed that their differences are more prominent when the number of spins  $N = 8$ . For this case, the two methods have comparable relative energy error (except at  $J/h = 1.0$ , when Method-I has higher relative error) but Method-I takes less time to converge.

We admit that the system sizes we considered are small and not significant in establishing the credibility of NQS in finding excited states. For higher values of  $N$ , as well as  $J/h$ , our preliminary runs showed that the system was not able to converge in a reasonable number of epochs and had high relative error values after exhausting the maximum number of epochs. We discuss more on this in Sec. 5.2.

# CHAPTER 5

## Conclusion and Future Work

In this chapter, we summarise the contributions made in Sec. 5.1 and discuss possible threads for future work in Sec. 5.2.

### 5.1 Conclusion

In this work we evaluated classical machine learning models, specifically Multilayer Perceptron (MLP) and restricted Boltzmann machines (RBM), in their ability to capture the inherent characteristics of two quantum many-body models, namely (i) the one-dimensional Ising  $J_1 - J_2$  model for determining its ground state and (ii) the one-dimensional Ising model for determining excited states.

For the first part, we specifically evaluated whether transfer learning improves the scalability, efficiency and effectiveness of neural quantum states, for various phases of the Ising  $J_1 - J_2$  system. We leveraged the physics-inspired transfer learning protocols,  $(k, p)$ -tiling, proposed by Zen et al. in [102], when determining the ground state. For these experiments, we started with a trained neural quantum state network for a given small size system ( $N = 4$  in our experiments). We then chose  $k$  and  $p$  values for the protocol to transfer weights to a larger system that was  $p$  times the base system's size. We evaluated these protocols for the RBM and MLP architectures, and concluded in Sec. 3.2 that the  $(L, p)$ -tiling protocol was the most effective and efficient, particularly for systems with larger size.

Secondly, we evaluated two methods to determine low-lying excited states in the one-dimensional Ising model. The first of the two uses an orthogonal penalty term to modify the Hamiltonian and was detailed in Sec. 4.1.1. The second method we proposed explicitly reconstructs the Hamiltonian such that the eigenvalues are rearranged, as shown in Sec. 4.1.2. We were able to evaluate the effectiveness of both these approaches for systems with  $N = \{4, 8\}$  spins for small values of  $J/h$ . While both methods were able to identify the first excited states effectively, Method-I was more efficient and this was more pronounced when  $N = 8$ . From the experiments reported we observed that the effectiveness suffered as we went to higher values of  $J/h$  within the same system size as well as to systems of higher size. We believe that the samples generated from Monte Carlo sampling may not

be representative enough to capture the intricacies of large sized systems, resulting in the neural network being unable to find excited states for such systems.

It should also be noted that the implementation for this work relied heavily on the `nqs-tensorflow2` [100] library. Additionally, evaluating transfer learning across size, as well as parameters across system sizes, required setting up a pipeline (See ??) that worked with the NSCC job scheduler. Even with such a setup in place to automate tasks, the time taken for completion of one set of experiments (say, transfer learning across size through MLP using unsupervised learning) was of the order of a few weeks, being even longer for systems of larger size.

## 5.2 Future Work

Within this field of machine learning for quantum physics, there exist several interesting but challenging problems. One of the fundamental problems is indeed to find the ground state and in this thesis, we evaluated transfer learning protocols to do so for the Ising  $J_1 - J_2$  model.

Our evaluation of transfer learning protocols was aimed at measuring their scalability, effectiveness and efficiency as we increased system sizes. We acknowledge that studying phase transitions is an equally key problem to study. However, we table that for potential future work owing to the fine-grained experiments such a study requires, particularly from the standpoint of identifying critical points during transitions.

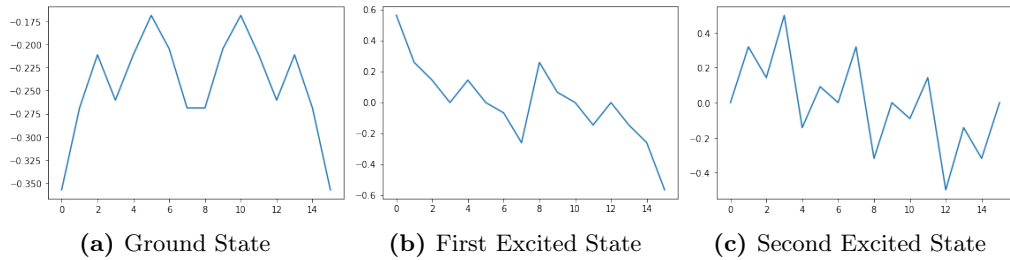
Note that, while evaluating the transfer learning protocols, our direction of transfer was from small sized systems to larger ones. This motivated the  $(k, p) - \text{tiling}$  such that  $k$  weights were repeated  $p$  times. The renormalisation group theory in theoretical physics [95] states that “the system at one scale will be seen to consist of self-similar copies of itself when viewed at a smaller scale, with different parameters describing the components of the system”. This implies that the system on a smaller scale would have different parameters than the one at larger scale. The idea of evaluating systems at different scales, particularly from large to small, resembles the renormalisation group theory. Evaluating this direction of transfer learning would require devising protocols that marginalise or truncate the parameters of the neural network quantum states, as opposed to the existing ones that replicate the parameters.

In the context of transfer learning, one could also study the connection between different quantum models. For instance, studying the connection of between the solution of different boundary conditions. Given that the connection between a model with open boundary conditions and one with closed boundary conditions can be described in terms of the Hamiltonian, it may be interesting to analyse if we can apply or transfer the solution of the former to the latter and vice-versa. This branch of inquiry could further be extended to the context of dimensions, such as evaluating the feasibility of using the solution for a

one-dimensional (16 spins) Ising model in a two-dimensional ( $4 \times 4$ ) Ising model. Lastly, studying whether transferring of parameters from one quantum model, say Heisenberg, to another quantum model, for instance, Ising, could also help deepen our understanding of how the two models are related to each other. The transfer learning protocols proposed by the authors of [102] could help empirically answer some of these questions.

In this work, we also attempted to find low-lying excited states using the complex MLP neural network quantum states. However, we could only identify a few of them with reasonable accuracy using the two approaches discussed in this work. Aside from the findings reported in Chapter 4, we conducted some preliminary experiments and observed that Method-I was more scalable as we were able to determine the first excited state for  $N = 16$ , while this was not possible for Method-II. Note that Method-II rearranges eigenvalues by explicitly constructing a Hamiltonian as shown in Eq. 4.22. The denominator in this calculation becomes a  $2^{16} \times 2^{16}$  matrix, which was intractable on our compute resources. Moreover, the Hamiltonian matrices constructed through Method-II are non-sparse matrices and obtaining their exact eigenvalues and eigenvectors becomes infeasible for  $N = 16$ .

In the ground state for the systems we've considered, the wave function has been symmetric, but this was not the case for the first and second excited states (See Fig. 5.1). This implies that as we go to higher excited states, the wave function gets more complicated, making it harder for our system to represent. Despite this, we were able to find the second excited



**Figure 5.1:** Exact wave function plots for  $N = 4$ ,  $J/h = 0.5$ . Note the asymmetry in the excited state wave functions as opposed to the symmetry in the ground state wave function

state for the one-dimensional Ising model with sizes 4 and 8, using Method-II, but not with Method-I. Note that in Method-I, for the second excited state the objective function includes the orthogonal penalty term which is the sum of the overlap of the second excited state with the ground state and the overlap of the second excited state with the first excited state. In our experiments, we observed that of these two terms only the former goes to zero, while the latter failed to go below 60%, causing the optimization to eventually diverge.

It should also be noted about excited states that as we go from the ground state to the higher excited states, the energy values tend to have smaller gap between them (See Fig. ?? in Appendix for reference). This increases the chances for the optimization to be stuck in a higher excited state, while we are trying for a low-lying excited state. In the ground state

exploration, we've observed how transfer learning is useful in preventing the optimization from getting stuck in a local minima. We're yet to explore transfer learning in the context of excited states. This could include transferring the ground state to get to the excited state, using the fact that an excited state must always be orthogonal to the ground state. Another transfer learning use-case in this context could be to transfer— samples or weights, from Method-II to Method-I, or vice-versa.

Recall that Method-I for finding low-lying excited states in Chapter 4 imposes a constraint on the the objective function. Aside from this constrained optimisation, an alternative way to impose constraints on neural-network quantum states is to constraint the sampling. There are two possible ways to introduce constraints in the sampling process. A possible way is that we generate samples without any constraints and then test the samples against a constraint, rejecting those that violate the constraint. Alternatively, we apply the constraint to the sample generation process itself. For instance, while sampling (say using Metropolis sampling) for the Heisenberg model the total number of spin-up and spin-down must be the same, to ensure zero magnetisation. This can be done by swapping two spins in a random position during the sampling process instead of flipping random spins to maintain the total number. Also note that when imposing constraints on the optimisation, they could be imposed on the weights, instead of being imposed on the objective function. If the weights are known to follow some distribution, or fit a particular pattern, they can be modified after the gradient is applied to obey the desired constraint. There exist two key symmetries in quantum systems: translational and rotational symmetry. The former exists in a system with periodic boundary conditions such that we can translate the system and it should be in the same state. Carleo and Troyer discuss this in [12] using constrained optimisation. Rotational symmetry, on the other hand, exists when the amplitude for spin-up and spin-down is the same. This is true for the one-dimensional Ising model, as the amplitude for all spin-up and all spin-down is the same therein. This could be imposed as a constraint during the sampling process.



## References

- [1] National supercomputing centre (nsc). <https://www.nsc.sg/>. [Online; accessed 08-Oct-2021].
- [2] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (2016), pp. 265–283.
- [3] AÎMEUR, E., BRASSARD, G., AND GAMBS, S. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence* (2006), Springer, pp. 431–442.
- [4] AMMAR, H. B., MOCANU, D. C., TAYLOR, M. E., DRIESSENS, K., TUYLS, K., AND WEISS, G. Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2013), Springer, pp. 449–464.
- [5] ARSENAULT, L.-F. M. C., LOPEZ-BEZANILLA, A., VON LILIENFELD, O. A., AND MILLIS, A. J. Machine learning for many-body physics: The case of the anderson impurity model. *Phys. Rev. B* 90 (Oct 2014), 155136.
- [6] BAKER, N., ALEXANDER, F., BREMER, T., HAGBERG, A., KEVREKIDIS, Y., NAJM, H., PARASHAR, M., PATRA, A., SETHIAN, J., WILD, S., ET AL. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Tech. rep., USDOE Office of Science (SC), Washington, DC (United States), 2019.
- [7] BAUER, B., CARR, L., EVERTZ, H. G., FEIGUIN, A., FREIRE, J., FUCHS, S., GAMPER, L., GUKELBERGER, J., GULL, E., GUERTLER, S., ET AL. The alps project release 2.0: open source software for strongly correlated systems. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 05 (2011), P05001.
- [8] BRADBURY, J., FROSTIG, R., HAWKINS, P., JOHNSON, M. J., LEARY, C., MACLAURIN, D., NECULA, G., PASZKE, A., VANDERPLAS, J., WANDERMAN-MILNE, S., AND ZHANG, Q. JAX: composable transformations of Python+NumPy programs, 2018.
- [9] CAI, Z., AND LIU, J. Approximating quantum many-body wave functions using artificial neural networks. *Phys. Rev. B* 97 (Jan 2018), 035116.

- [10] CARLEO, G., CHOO, K., HOFMANN, D., SMITH, J. E., WESTERHOUT, T., ALET, F., DAVIS, E. J., EFTHYMIU, S., GLASSER, I., LIN, S.-H., ET AL. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX* 10 (2019), 100311.
- [11] CARLEO, G., CIRAC, I., CRANMER, K., DAUDET, L., SCHULD, M., TISHBY, N., VOGT-MARANTO, L., AND ZDEBOROVÁ, L. Machine learning and the physical sciences. *Reviews of Modern Physics* 91, 4 (2019), 045002.
- [12] CARLEO, G., AND TROYER, M. Solving the quantum many-body problem with artificial neural networks. *Science* 355, 6325 (2017), 602–606.
- [13] CARRASQUILLA, J. Machine learning for quantum matter, 2020.
- [14] CARRASQUILLA, J., AND MELKO, R. G. Machine learning phases of matter. *Nature Physics* 13, 5 (2017), 431–434.
- [15] CARRASQUILLA, J., TORLAI, G., MELKO, R. G., AND AOLITA, L. Reconstructing quantum states with generative models. *Nature Machine Intelligence* 1, 3 (2019), 155–161.
- [16] CHEN, J., CHENG, S., XIE, H., WANG, L., AND XIANG, T. Equivalence of restricted boltzmann machines and tensor network states. *Physical Review B* 97, 8 (2018), 085104.
- [17] CHOO, K., CARLEO, G., REGNAULT, N., AND NEUPERT, T. Symmetries and many-body excitations with neural-network quantum states. *Physical review letters* 121, 16 (2018), 167204.
- [18] CHOO, K., MEZZACAPO, A., AND CARLEO, G. Fermionic neural-network states for ab-initio electronic structure. *Nature communications* 11, 1 (2020), 1–7.
- [19] CHOO, K., NEUPERT, T., AND CARLEO, G. Two-dimensional frustrated j 1- j 2 model studied with neural network quantum states. *Physical Review B* 100, 12 (2019), 125124.
- [20] COATES, A., NG, A., AND LEE, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 215–223.
- [21] COLLURA, M., DELL’ANNA, L., FELSER, T., AND MONTANGERO, S. On the descriptive power of neural-networks as constrained tensor networks with exponentially large bond dimension. *arXiv preprint arXiv:1905.11351* (2019).
- [22] COLOMBET, I., RUELLAND, A., CHATELLIER, G., GUEYFFIER, F., DEGOULET, P., AND JAULENT, M.-C. Models to predict cardiovascular risk: comparison of cart, multilayer perceptron and logistic regression. In *Proceedings of the AMIA Symposium* (2000), American Medical Informatics Association, p. 156.

- [23] CRONE, S. F., AND KOEPEL, C. Predicting exchange rates with sentiment indicators: An empirical evaluation using text mining and multilayer perceptrons. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFER)* (2014), IEEE, pp. 114–121.
- [24] DAHL, G., RANZATO, M., MOHAMED, A.-R., AND HINTON, G. E. Phone recognition with the mean-covariance restricted boltzmann machine. *Advances in neural information processing systems 23* (2010), 469–477.
- [25] DENG, D.-L., LI, X., AND SARMA, S. D. Machine learning topological states. *Physical Review B* 96, 19 (2017), 195145.
- [26] DENG, D.-L., LI, X., AND SARMA, S. D. Quantum entanglement in neural network states. *Physical Review X* 7, 2 (2017), 021021.
- [27] DUNJKO, V., TAYLOR, J. M., AND BRIEGEL, H. J. Quantum-enhanced machine learning. *Physical review letters* 117, 13 (2016), 130501.
- [28] EISERT, J., CRAMER, M., AND PLENIO, M. B. Colloquium: Area laws for the entanglement entropy. *Reviews of modern physics* 82, 1 (2010), 277.
- [29] FEHSKE, H., SCHNEIDER, R., AND WEISSE, A. *Computational many-particle physics*, vol. 739. Springer, 2007.
- [30] FIORE, U., PALMIERI, F., CASTIGLIONE, A., AND DE SANTIS, A. Network anomaly detection with the restricted boltzmann machine. *Neurocomputing* 122 (2013), 13–23.
- [31] FISHMAN, M., WHITE, S. R., AND SToudenMIRE, E. M. The itensor software library for tensor network calculations. *arXiv preprint arXiv:2007.14822* (2020).
- [32] GAO, X., AND DUAN, L.-M. Efficient representation of quantum many-body states with deep neural networks. *Nature communications* 8, 1 (2017), 1–6.
- [33] GARDNER, M. W., AND DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.
- [34] GHIRINGHELLI, L. M., VYBIRAL, J., LEVCHENKO, S. V., DRAXL, C., AND SCHEFFLER, M. Big data of materials science: critical role of the descriptor. *Physical review letters* 114, 10 (2015), 105503.
- [35] GLASSER, I., PANCOTTI, N., AUGUST, M., RODRIGUEZ, I. D., AND CIRAC, J. I. Neural-network quantum states, string-bond states, and chiral topological states. *Physical Review X* 8, 1 (2018), 011006.
- [36] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

- [37] HENDRY, D., AND FEIGUIN, A. E. Machine learning approach to dynamical properties of quantum many-body systems. *Physical Review B* 100, 24 (2019), 245123.
- [38] HIBAT-ALLAH, M., GANAHL, M., HAYWARD, L. E., MELKO, R. G., AND CARRASQUILLA, J. Recurrent neural network wave functions. *Physical Review Research* 2, 2 (2020), 023358.
- [39] HINTON, G. E., AND SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.
- [40] HINTON, G. E., AND SALAKHUTDINOV, R. R. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems* (2009), pp. 1607–1614.
- [41] JAITLEY, N., AND HINTON, G. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), IEEE, pp. 5884–5887.
- [42] JIA, Z., YI, B., ZHAI, R., WU, Y., GUO, G., AND GUO, G. Quantum neural network states: A brief review of methods and applications. *Advanced Quantum Technologies* 2, 7-8 (Mar 2019), 1800077.
- [43] JIA, Z.-A., ZHANG, Y.-H., WU, Y.-C., KONG, L., GUO, G.-C., AND GUO, G.-P. Efficient machine-learning representations of a surface code with boundaries, defects, domain walls, and twists. *Physical Review A* 99, 1 (2019), 012307.
- [44] KALININ, S. V., SUMPTER, B. G., AND ARCHIBALD, R. K. Big-deep-smart data in imaging for guiding materials design. *Nature materials* 14, 10 (2015), 973–980.
- [45] KAO, Y.-J., HSIEH, Y.-D., AND CHEN, P. Uni10: An open-source library for tensor network algorithms. In *Journal of Physics: Conference Series* (2015), vol. 640, IOP Publishing, p. 012040.
- [46] KHAN, A., BAHARUDIN, B., LEE, L. H., AND KHAN, K. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology* 1, 1 (2010), 4–20.
- [47] KIM, J., BACZEWSKI, A. D., BEAUDET, T. D., BENALI, A., BENNETT, M. C., BERRILL, M. A., BLUNT, N. S., BORDA, E. J. L., CASULA, M., CEPERLEY, D. M., ET AL. Qmcpack: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids. *Journal of Physics: Condensed Matter* 30, 19 (2018), 195901.
- [48] KUMAR, M., AND YADAV, N. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. *Computers & Mathematics with Applications* 62, 10 (2011), 3796–3811.

- [49] KUSNE, A. G., GAO, T., MEHTA, A., KE, L., NGUYEN, M. C., HO, K.-M., ANTROPOV, V., WANG, C.-Z., KRAMER, M. J., LONG, C., ET AL. On-the-fly machine-learning for high-throughput experiments: search for rare-earth-free permanent magnets. *Scientific reports* 4, 1 (2014), 1–7.
- [50] LAGARIS, I. E., LIKAS, A., AND FOTIADIS, D. I. Artificial neural network methods in quantum mechanics. *Computer Physics Communications* 104, 1-3 (1997), 1–14.
- [51] LAROCHELLE, H., AND BENGIO, Y. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning* (2008), pp. 536–543.
- [52] LEGGETT, A. J. *Quantum liquids: Bose condensation and Cooper pairing in condensed-matter systems*. Oxford Univ. Press, 2015.
- [53] LIAO, L., JIN, W., AND PAVEL, R. Enhanced restricted boltzmann machine with prognosability regularization for prognostics and health assessment. *IEEE Transactions on Industrial Electronics* 63, 11 (2016), 7076–7083.
- [54] LIU, W., WANG, Z., LIU, X., ZENG, N., LIU, Y., AND ALSAADI, F. E. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (2017), 11–26.
- [55] LONG, M. D. H. *DEEP QUANTUM*. Bachelor’s thesis, 2019.
- [56] LU, N., LI, T., REN, X., AND MIAO, H. A deep learning scheme for motor imagery classification based on restricted boltzmann machines. *IEEE transactions on neural systems and rehabilitation engineering* 25, 6 (2016), 566–576.
- [57] LU, S., GAO, X., AND DUAN, L.-M. Efficient representation of topologically ordered states with restricted boltzmann machines. *Physical Review B* 99, 15 (2019), 155136.
- [58] LUO, D., AND CLARK, B. K. Backflow transformations via neural networks for quantum many-body wave functions. *Physical review letters* 122, 22 (2019), 226401.
- [59] MEHTA, P., AND SCHWAB, D. J. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831* (2014).
- [60] MELKO, R. G., CARLEO, G., CARRASQUILLA, J., AND CIRAC, J. I. Restricted boltzmann machines in quantum physics. *Nature Physics* 15, 9 (2019), 887–892.
- [61] MIDHUN, M., NAIR, S. R., PRABHAKAR, V. N., AND KUMAR, S. S. Deep model for classification of hyperspectral image using restricted boltzmann machine. In *Proceedings of the 2014 international conference on interdisciplinary advances in applied computing* (2014), pp. 1–7.
- [62] MIN, Y. Approximating excited states using neural networks. *arXiv preprint arXiv:2012.13268* (2020).

- 
- [63] MURTAGH, F. Multilayer perceptrons for classification and regression. *Neurocomputing* 2, 5-6 (1991), 183–197.
- [64] NEEDS, R., TOWLER, M., DRUMMOND, N., LOPEZ RIOS, P., AND TRAIL, J. Variational and diffusion quantum monte carlo calculations with the casino code. *The Journal of chemical physics* 152, 15 (2020), 154106.
- [65] NGUYEN, T. D., PHUNG, D. Q., HUYNH, V., AND LE, T. Supervised restricted boltzmann machines. In *UAI* (2017).
- [66] NOMURA, Y. Machine learning quantum states—extensions to fermion–boson coupled systems and excited-state calculations. *Journal of the Physical Society of Japan* 89, 5 (2020), 054706.
- [67] NOMURA, Y. Helping restricted boltzmann machines with quantum-state representation by restoring symmetry. *Journal of Physics: Condensed Matter* 33, 17 (2021), 174003.
- [68] NOMURA, Y., DARMAWAN, A. S., YAMAJI, Y., AND IMADA, M. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B* 96, 20 (2017), 205152.
- [69] NOMURA, Y., AND IMADA, M. Dirac-type nodal spin liquid revealed by machine learning. *arXiv preprint arXiv:2005.14142* (2020).
- [70] ORHAN, U., HEKIM, M., AND OZER, M. Eeg signals classification using the k-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications* 38, 10 (2011), 13475–13481.
- [71] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.
- [72] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., ET AL. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.
- [73] PFAU, D., SPENCER, J. S., MATTHEWS, A. G., AND FOULKES, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research* 2, 3 (2020), 033429.
- [74] PLENIO, M. B., EISERT, J., DREISSIG, J., AND CRAMER, M. Entropy, entanglement, and area: analytical results for harmonic lattice systems. *Physical review letters* 94, 6 (2005), 060503.

- 
- [75] ROBERTS, C., MILSTED, A., GANAHL, M., ZALCMAN, A., FONTAINE, B., ZOU, Y., HIDARY, J., VIDAL, G., AND LEICHENAUER, S. Tensornetwork: A library for physics and machine learning. *arXiv preprint arXiv:1905.01330* (2019).
- [76] RUCK, D. W., ROGERS, S. K., AND KABRISKY, M. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing* 2, 2 (1990), 40–48.
- [77] SAITO, H. Solving the bose–hubbard model with machine learning. *Journal of the Physical Society of Japan* 86, 9 (Sep 2017), 093001.
- [78] SAITO, H. Method to solve quantum few-body problems with artificial neural networks. *Journal of the Physical Society of Japan* 87, 7 (2018), 074002.
- [79] SAITO, H., AND KATO, M. Machine learning technique to find quantum many-body ground states of bosons on a lattice. *Journal of the Physical Society of Japan* 87, 1 (2018), 014001.
- [80] SANDVIK, A. W. Finite-size scaling of the ground-state parameters of the two-dimensional heisenberg model. *Physical Review B* 56, 18 (1997), 11678.
- [81] SARMA, S. D., DENG, D.-L., AND DUAN, L.-M. Machine learning meets quantum physics. *arXiv preprint arXiv:1903.03516* (2019).
- [82] SCHOENHOLZ, S. S., CUBUK, E. D., SUSSMAN, D. M., KAXIRAS, E., AND LIU, A. J. A structural approach to relaxation in glassy liquids. *Nature Physics* 12, 5 (2016), 469–471.
- [83] SHARIR, O., LEVINE, Y., WIES, N., CARLEO, G., AND SHASHUA, A. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical review letters* 124, 2 (2020), 020503.
- [84] TANG, Y., SALAKHUTDINOV, R., AND HINTON, G. Robust boltzmann machines for recognition and denoising. In *2012 IEEE conference on computer vision and pattern recognition* (2012), IEEE, pp. 2264–2271.
- [85] TEH, Y. W., AND HINTON, G. E. Rate-coded restricted boltzmann machines for face recognition. In *Advances in neural information processing systems* (2001), pp. 908–914.
- [86] TENG, P. Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks. *Physical Review E* 98, 3 (2018), 033305.
- [87] TOMCZAK, J. M., AND ZIĘBA, M. Classification restricted boltzmann machine for comprehensible credit scoring model. *Expert Systems with Applications* 42, 4 (2015), 1789–1796.
- [88] TORLAI, G., AND MELKO, R. G. Neural decoder for topological codes. *Physical review letters* 119, 3 (2017), 030501.

- [89] TORLAI, G., AND MELKO, R. G. Machine-learning quantum states in the nisq era. *Annual Review of Condensed Matter Physics* 11 (2020), 325–344.
- [90] ĐURIĆ, T., AND ŠEVA, T. Efficient neural-network based variational monte carlo scheme for direct optimization of excited energy states in frustrated quantum systems. *Physical Review B* 102, 8 (2020), 085104.
- [91] VANKAYALA, V. S. S., AND RAO, N. D. Artificial neural networks and their applications to power systems—a bibliographical survey. *Electric power systems research* 28, 1 (1993), 67–79.
- [92] WEI, B., AND PAL, C. Heterogeneous transfer learning with rbms. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2011), vol. 25.
- [93] WEISS, K., KHOSHGOFTAAR, T. M., AND WANG, D. A survey of transfer learning. *Journal of Big data* 3, 1 (2016), 9.
- [94] WESTERHOUT, T., ASTRAKHANTSEV, N., TIKHONOV, K. S., KATSNELSON, M. I., AND BAGROV, A. A. Generalization properties of neural network approximations to frustrated magnet ground states. *Nature Communications* 11, 1 (Mar 2020).
- [95] WILSON, K. G. Problems in physics with many scales of length. *Scientific American* 241, 2 (1979), 158–179.
- [96] WU, Y., DUAN, L.-M., AND DENG, D.-L. Artificial neural network based computation for out-of-time-ordered correlators. *Physical Review B* 101, 21 (2020), 214308.
- [97] XIE, P., DENG, Y., AND XING, E. Diversifying restricted boltzmann machine for document modeling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), pp. 1315–1324.
- [98] YANG, L., HU, W., AND LI, L. Scalable variational monte carlo with graph neural ansatz. *arXiv preprint arXiv:2011.12453* (2020).
- [99] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (2014), Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc., pp. 3320–3328.
- [100] ZEN, R. nqs-tensorflow2. <https://github.com/remmyzen/nqs-tensorflow2>. [Online; accessed 08-Oct-2021].
- [101] ZEN, R., MY, L., TAN, R., HÉBERT, F., GATTOBIGIO, M., MINIATURA, C., POLETTI, D., AND BRESSAN, S. Finding quantum critical points with neural-network quantum states. *arXiv preprint arXiv:2002.02618* (2020).



- 
- [102] ZEN, R., MY, L., TAN, R., HÉBERT, F., GATTOBIGIO, M., MINIATURA, C., POLETTI, D., AND BRESSAN, S. Transfer learning for scalability of neural-network quantum states. *Physical Review E* 101, 5 (2020), 053301.
  - [103] ZHANG, J. Deep transfer learning via restricted boltzmann machine for document classification. In *2011 10th International Conference on Machine Learning and Applications and Workshops* (2011), vol. 1, IEEE, pp. 323–326.

# CHAPTER 6

## Appendix

### 6.1 Example Hamiltonian Matrix

As an example, consider the calculation of a one-dimensional Ising model with 3 particles ( $N = 3$ ) and periodic boundary conditions. Therefore, there are  $2^N = 2^3 = 8$  possible configurations corresponding to a basis vector  $|x\rangle = |x_1, x_2, x_3\rangle$  listed below

- $|\uparrow\uparrow\uparrow\rangle$
- $|\uparrow\downarrow\uparrow\rangle$
- $|\downarrow\uparrow\uparrow\rangle$
- $|\downarrow\downarrow\uparrow\rangle$
- $|\uparrow\uparrow\downarrow\rangle$
- $|\uparrow\downarrow\downarrow\rangle$
- $|\downarrow\uparrow\downarrow\rangle$
- $|\downarrow\downarrow\downarrow\rangle$

For each of these, the corresponding basis vector is obtained by leveraging the Kronecker product<sup>1</sup>, denoted by the  $\otimes$  symbol. Note that the corresponding vector for  $\uparrow$  is:  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ , while that for  $\downarrow$  is:  $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ . Therefore,

$$|\uparrow\downarrow\uparrow\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Kronecker\\_product](https://en.wikipedia.org/wiki/Kronecker_product)

Similarly, for each of the remaining 7 spin groups we have:

$$\begin{aligned}
 |\uparrow\uparrow\uparrow\rangle &\equiv \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |\uparrow\uparrow\downarrow\rangle &\equiv \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |\uparrow\downarrow\downarrow\rangle &\equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
 |\downarrow\uparrow\uparrow\rangle &\equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |\downarrow\uparrow\downarrow\rangle &\equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & |\downarrow\downarrow\uparrow\rangle &\equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & |\downarrow\downarrow\downarrow\rangle &\equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
 \end{aligned}$$

Now that we know the vector representation for each of the configurations, we can determine the spin operators for a given model. Next, we consider the Ising model, and populate its Hamiltonian matrix using these configurations.

### Ising Model

Expanding upon the Hamiltonian given in Eq. 2.5, we can write,

$$\begin{aligned}
 H_I &= -h(\sigma_1^x + \sigma_2^x + \sigma_3^x) - J(\sigma_1^z \sigma_2^z + \sigma_2^z \sigma_3^z + \sigma_3^z \sigma_1^z) \\
 &= \begin{bmatrix} -3J & -h & -h & 0 & -h & 0 & 0 & 0 \\ -h & J & 0 & -h & 0 & -h & 0 & 0 \\ -h & 0 & J & -h & 0 & 0 & -h & 0 \\ 0 & -h & -h & J & 0 & 0 & 0 & -h \\ -h & 0 & 0 & 0 & J & -h & -h & 0 \\ 0 & -h & 0 & 0 & -h & J & 0 & -h \\ 0 & 0 & -h & 0 & -h & 0 & J & -h \\ 0 & 0 & 0 & -h & 0 & -h & -h & -3J \end{bmatrix} \quad (6.1)
 \end{aligned}$$

The result of multiplying various Pauli matrices with a basis state for the this context is given in Eq. 6.2. It can be extended to basis states with more than one elements as well— $\sigma_2^x |\uparrow\uparrow\uparrow\rangle = |\uparrow\downarrow\uparrow\rangle$  or  $\sigma_3^z |\downarrow\uparrow\downarrow\rangle = -|\downarrow\uparrow\downarrow\rangle$ .

$$\begin{aligned}
 \sigma^z |\uparrow\rangle &= |\uparrow\rangle & \sigma^z |\downarrow\rangle &= |\downarrow\rangle \\
 \sigma^x |\uparrow\rangle &= |\downarrow\rangle & \sigma^x |\downarrow\rangle &= -|\uparrow\rangle
 \end{aligned} \quad (6.2)$$

In order to find the matrix representation of the Hamiltonian, let  $H$  act upon each of the basis vectors. Consider the first basis vector, corresponding to  $|\uparrow\uparrow\uparrow\rangle$ ,

$$H_I |\uparrow\uparrow\uparrow\rangle = -J(\sigma_1^z \sigma_2^z |\uparrow\uparrow\uparrow\rangle + \sigma_2^z \sigma_3^z |\uparrow\uparrow\uparrow\rangle + \sigma_3^z \sigma_1^z |\uparrow\uparrow\uparrow\rangle) - h(\sigma_1^x |\uparrow\uparrow\uparrow\rangle + \sigma_2^x |\uparrow\uparrow\uparrow\rangle + \sigma_3^x |\uparrow\uparrow\uparrow\rangle)$$

Consider the first term in the coefficients of the interaction constant, i.e.,  $\sigma_1^z \sigma_2^z |\uparrow\uparrow\uparrow\rangle$ .

- The easier way to compute the result of this interaction is to take into account the action of the spin operators as explained in Eq. 6.2, with the subscripts in the  $\sigma^z$  values indicating which of the spins to interact with, i.e.,  $\sigma_1^z$  acts upon the first  $\uparrow$ , while  $\sigma_2^z$  with the second and so on. This would result in  $|\uparrow\uparrow\uparrow\rangle$ .
- On the other hand, the more mathematical computation for the same operation is to first take the Kronecker product of the two sigma values and simultaneously a Kronecker product of the spins. Once the Kronecker product from each of these is obtained, we carry out matrix multiplication between them. The result would still be the same, except that it would be in the basis vector form of  $|\uparrow\uparrow\uparrow\rangle$ .

Therefore, we get

$$\begin{aligned}
 H_I |\uparrow\uparrow\uparrow\rangle &= -J(|\uparrow\uparrow\uparrow\rangle + |\uparrow\uparrow\uparrow\rangle + |\uparrow\uparrow\uparrow\rangle) - h(|\downarrow\uparrow\uparrow\rangle + |\uparrow\downarrow\uparrow\rangle + |\uparrow\uparrow\downarrow\rangle) \\
 &= -3J(|\uparrow\uparrow\uparrow\rangle) - h(|\downarrow\uparrow\uparrow\rangle + |\uparrow\downarrow\uparrow\rangle + |\uparrow\uparrow\downarrow\rangle) \\
 H_I |\uparrow\uparrow\uparrow\rangle &= -3J \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - h \left( \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} -3J \\ -h \\ -h \\ 0 \\ -h \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (6.3)
 \end{aligned}$$

Thus, we have the first column of the Hamiltonian matrix, from the interaction of  $H$  with the first basis vector. Similarly, we arrive at the other columns of the Hamiltonian matrix from the other basis vectors. Finally,

$$H_I \equiv \begin{bmatrix} -3J & -h & -h & 0 & -h & 0 & 0 & 0 \\ -h & J & 0 & -h & 0 & -h & 0 & 0 \\ -h & 0 & J & -h & 0 & 0 & -h & 0 \\ 0 & -h & -h & J & 0 & 0 & 0 & -h \\ -h & 0 & 0 & 0 & J & -h & -h & 0 \\ 0 & -h & 0 & 0 & -h & J & 0 & -h \\ 0 & 0 & -h & 0 & -h & 0 & J & -h \\ 0 & 0 & 0 & -h & 0 & -h & -h & -3J \end{bmatrix} \quad (6.4)$$

From Eqn. 6.4,

$$H_{x,x'} = \begin{cases} -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j & \text{if } x = x' = |\sigma_1 \sigma_2 \cdots \sigma_N\rangle \\ -h & \text{if } x \text{ and } x' \text{ differ only at one spin} \\ 0 & \text{otherwise} \end{cases} \quad (6.5)$$

## 6.2 Expression for $E_{loc}(x)$

In this section, the algebraic transformation that helps one arrive at the expression for  $E_{loc}(x)$  is shown. This has been reproduced here from [55] for completeness.

In quantum theory, the average energy of the system in state vector  $|\Psi\rangle$  is defined by

$$E[\Psi] = \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle} \quad (6.6)$$

From the variational principle of quantum physics, we know  $E[\Psi] \geq E_0$ , for all  $|\Psi\rangle$ 's, where  $E_0$  is the ground energy of the system.

By using the vector representation of  $|\Psi\rangle$  and the matrix representation of  $H$ , we have

$$\begin{aligned} \langle \Psi | \Psi \rangle &\triangleq \begin{bmatrix} \psi^*(\mathbf{x}_1) & \psi^*(\mathbf{x}_2) & \dots & \psi^*(\mathbf{x}_d) \end{bmatrix} \begin{bmatrix} \psi(\mathbf{x}_1) \\ \psi(\mathbf{x}_2) \\ \vdots \\ \psi(\mathbf{x}_d) \end{bmatrix} \\ &= \psi^*(\mathbf{x}_1)\psi(\mathbf{x}_1) + \psi^*(\mathbf{x}_2)\psi(\mathbf{x}_2) + \dots + \psi^*(\mathbf{x}_d)\psi(\mathbf{x}_d) \\ &= \sum_{\mathbf{x} \in \{\mathbf{x}_1, \dots, \mathbf{x}_d\}} |\psi(\mathbf{x})|^2 \end{aligned} \quad (6.7)$$

$$\begin{aligned} \text{and, } \langle \Psi | H | \Psi \rangle &\triangleq \begin{bmatrix} \psi^*(\mathbf{x}_1) & \psi^*(\mathbf{x}_2) & \dots & \psi^*(\mathbf{x}_d) \end{bmatrix} \begin{bmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,d} \\ H_{2,1} & H_{2,2} & \dots & H_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ H_{d,1} & H_{d,2} & \dots & H_{d,d} \end{bmatrix} \begin{bmatrix} \psi(\mathbf{x}_1) \\ \psi(\mathbf{x}_2) \\ \vdots \\ \psi(\mathbf{x}_d) \end{bmatrix} \\ &= \sum_{\mathbf{x} \in \{\mathbf{x}_1, \dots, \mathbf{x}_d\}} \psi^*(\mathbf{x}) \left( \sum_{\mathbf{x}' \in \{\mathbf{x}_1, \dots, \mathbf{x}_d\}} H_{\mathbf{x}, \mathbf{x}'} \psi(\mathbf{x}') \right) \\ &= \sum_{\mathbf{x}, \mathbf{x}'} \psi^*(\mathbf{x}) H_{\mathbf{x}, \mathbf{x}'} \psi(\mathbf{x}') \end{aligned} \quad (6.8)$$

Substituting these in Eq. 6.6, the energy of the system in state  $|\Psi\rangle$  can be written as

$$E[\Psi] = \frac{\sum_{\mathbf{x}, \mathbf{x}'} \psi^*(\mathbf{x}) H_{\mathbf{x}, \mathbf{x}'} \psi(\mathbf{x}')}{\sum_{\mathbf{x} \in \{\mathbf{x}_1, \dots, \mathbf{x}_d\}} |\psi(\mathbf{x})|^2}$$

Note that the denominator is a summation over  $\mathbf{x}$ , resulting in a real number. For convenience,

let this real number be  $Z$ . Now,  $E[\Psi]$  can be rewritten as,

$$E[\Psi] = \sum_{\mathbf{x}, \mathbf{x}'} \frac{\psi^*(\mathbf{x})}{Z} H_{\mathbf{x}, \mathbf{x}'} \psi(\mathbf{x}') \quad (6.9)$$

Now we perform some algebraic transformations on  $E[\Psi]$  as follows:

$$\begin{aligned} E[\Psi] &= \sum_{\mathbf{x}, \mathbf{x}'} \frac{\psi^*(\mathbf{x})}{Z} H_{\mathbf{x}, \mathbf{x}'} \psi(\mathbf{x}') \\ &= \sum_{\mathbf{x}, \mathbf{x}'} \frac{\psi^*(\mathbf{x}) \psi(\mathbf{x})}{Z} H_{\mathbf{x}, \mathbf{x}'} \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x})} && \text{Multiply and divide by } \psi(\mathbf{x}) \\ &= \sum_{\mathbf{x}} \frac{|\psi(\mathbf{x})|^2}{Z} \left( \sum_{\mathbf{x}'} H_{\mathbf{x}, \mathbf{x}'} \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x})} \right) && \text{Separate summation over } \mathbf{x} \text{ and } \mathbf{x}' \end{aligned} \quad (6.10)$$

The quantity within the braces is a summation over  $\mathbf{x}'$  and depends only on  $\mathbf{x}$ . Let this quantity be called  $E_{loc}(\mathbf{x})$ , the local energy function,

$$E_{loc}(\mathbf{x}) = \sum_{\mathbf{x}'} H_{\mathbf{x}, \mathbf{x}'} \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x})}$$

The information of the Hamiltonian is now absorbed into this local energy function  $E_{loc}(\mathbf{x})$ .

### 6.3 Transfer Learning across parameters for Ising $J_1 - J_2$ model

To evaluate transfer learning across parameters for the Ising  $J_1 - J_2$  model, weights are transferred across various  $J_2$  values of the model within the same system size. We ranged the  $J_2$  values from  $[-3.0, -2.9, -2.8, \dots, 0]$ , *i.e.*, at increments of 0.1, while keeping  $J_1 = -0.5$  and  $h = 1$ . This set of experiments was conducted only for small system sizes, *i.e.*,  $\{4, 8, 16\}$  and in the unsupervised learning setting. Note that the number of weights stays the same in this case as there is no change in the size of the system across transfers. Therefore, no tiling protocol is needed. For transfer, the target system (the one with the higher  $J_2$  value) is initialised with weights from the converged model corresponding to the  $J_2$  value being transferred from. For instance, for transferring weights from a system with  $J_2 = -0.9$  to another with  $J_2 = -1.0$ , the latter will be initialised with the weights from the former at convergence and then fine-tuned. This implies that the time to convergence for the transfer learning run for  $J_2 = -1.0$  will include the time to convergence for all previous systems leading up to  $J_2 = -1.0$ , *i.e.*, the time taken by the system with  $J_2 = -3.0$ , plus that for system with  $J_2 = -2.9, \dots, J_2 = -1.1, J_2 = -1.0$  and finally the fine-tuning time that the system with  $J_2 = -1.0$  itself takes.

As with the transfer learning process described in Sec. 3.2, we use an iterative process for these experiments too. At every iteration,  $10^4$  samples are drawn (Gibbs sampling for RBM,

Metropolis algorithm with exchange strategy for MLP) using to evaluate the energy  $\langle H \rangle_\Psi$  and its gradients using the parameters of the RBM or MLP neural network being evaluated. Next, the parameters of the neural networks are updated using **RMSprop** optimiser to benefit from adaptive learning. The initial learning rate was set to 0.001. From our simulations, the activation function **ReLU** with one hidden layer containing thrice as many hidden nodes as the number of visible nodes performed well for the MLP and the same worked well for the RBM too. Additionally, the entries of the weight matrix  $\mathbf{W}$  that had to be set randomly during cold-start and transfer learning runs for the RBM were sampled from the Normal distribution,  $N(0, 0.01)$ , i.e., with zero mean and standard deviation 0.01. While training, a dynamic stopping criterion was used similar to the scalability case (Sec. 3.2). The threshold for standard deviation of local energy  $E_{loc}$  was set to be 0.005 and the maximum number of epochs were set to  $3 \times 10^4$ .

The experiments in this section were also run on an NVIDIA DGX-1 server equipped with NVIDIA Tesla V100 graphics processing units with 460 tensor cores, 5120 CUDA cores, and 16GB memory performed on the infrastructure of Singapore National Supercomputing Centre [1].

### 6.3.1 Transfer Learning across parameters using RBM

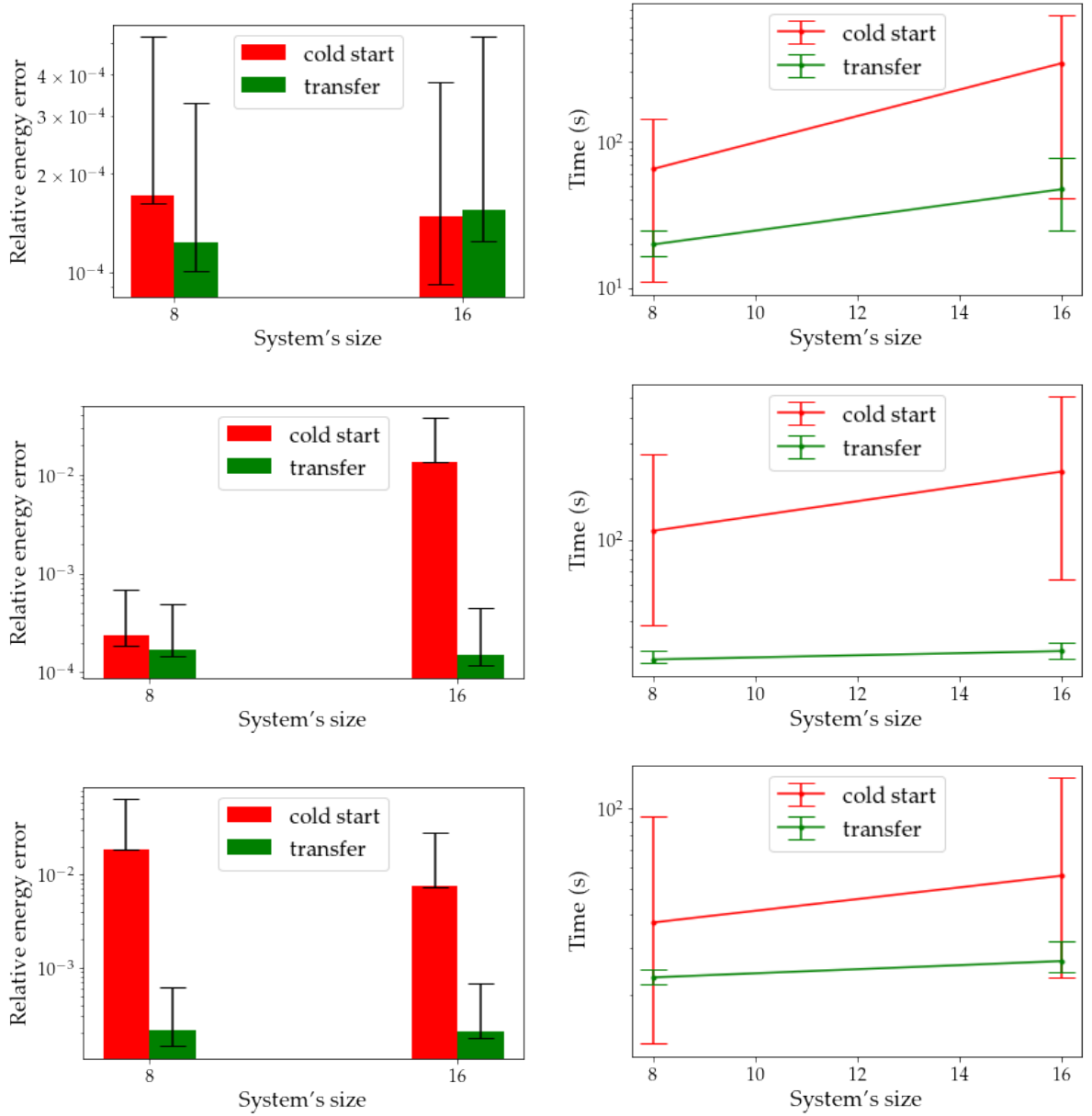
We first look at the effectiveness,  $\Delta E/E_0$  and efficiency of the transfer learning protocol against the cold-start one for the Ising  $J_1 - J_2$  model on employing the RBM architecture. We report these values across different values of  $J_2$  for systems with  $N = 8$  and  $N = 16$ , shown in Fig. 6.1.

We observe that even though the evaluated systems are of small size, the transfer learning protocol has a lower relative energy error, as well as smaller time to convergence. The error bars in the charts indicate the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, across 20 iterations. It can also be seen from the efficiency plots in Fig. 6.1 that the cold-start run has comparably longer error bars indicating varying latency across the iterations.

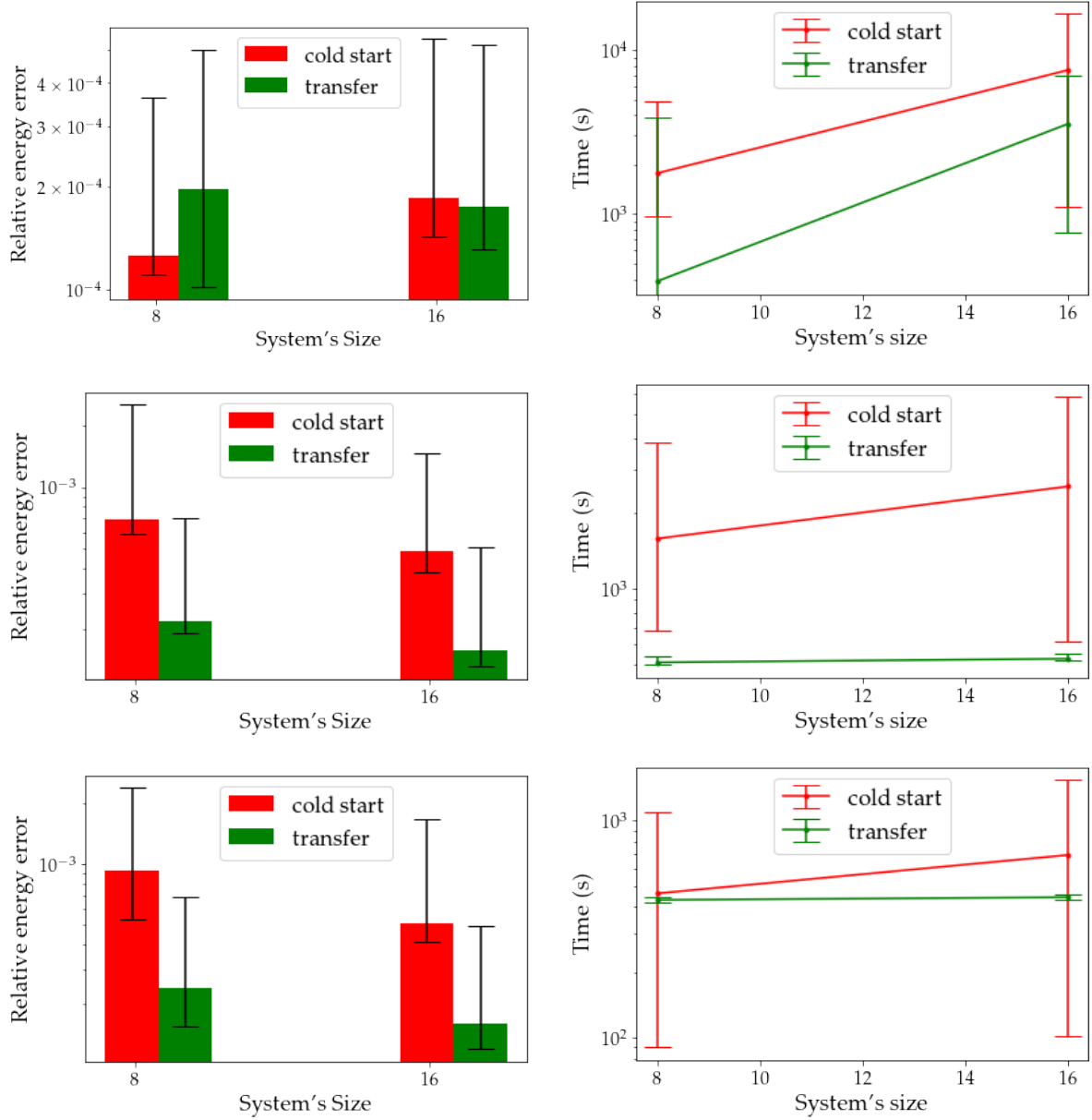
### 6.3.2 Transfer Learning across parameters using MLP

Fig. 6.2 plots the effectiveness and efficiency curves for the transfer across parameters for the Ising  $J_1 - J_2$  model, for sizes 8 and 16 when using the MLP architecture. Again, we can see that the transfer learning protocol shows reports relative error as well as shorter time to convergence, for most of the runs. Again the error bars in case of the efficiency plots are longer for the cold-start runs indicating that it is less stable across multiple iterations.





**Figure 6.1:** Effectiveness (left column) and Efficiency (right column) among cold-start and transfer learning runs, for systems with  $N = 8$  and  $N = 16$ , with  $h = 1.0$ ,  $J_1 = -0.5$  and  $J_2 = \{-1.0, -2.0, -3.0\}$  for Restricted Boltzmann Machine. The red bars and lines indicate cold-start, while the green ones indicate transfer learning runs. The error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, across 20 iterations.



**Figure 6.2:** Effectiveness (left column) and Efficiency (right column) among cold-start and transfer learning runs, for systems with  $N = 8$  and  $N = 16$ , with  $h = 1.0$ ,  $J_1 = -0.5$  and  $J_2 = \{-1.0, -2.0, -3.0\}$  for Multilayer Perceptron. The red bars and lines indicate cold-start, while the green ones indicate transfer learning runs. The error bars represent the interval of the 9<sup>th</sup> percentile and the 91<sup>st</sup> percentile, across 20 iterations.