Machine learning and quantum phases of matter

Hugo Théveniaut

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue le 24 novembre 2020 par :

Hugo THÉVENIAUT

Méthodes d'apprentissage automatique et phases quantiques de la matière

---

Ecole doctorale : **SDM - SCIENCES DE LA MATIERE - Toulouse**

Spécialité : **Physique**

Unité de recherche :
LPT-IRSAMC - Laboratoire de Physique Théorique

Thèse dirigée par
Fabien ALET

Jury

M. Nicolas REGNAULT, Rapporteur
M. Simon TREBST, Rapporteur
M. David GUéRY-ODELIN, Examinateur
M. Evert VAN NIEUWENBURG, Examinateur
Mme Eliska GREPLOVA, Examinatrice
M. Fabien ALET, Directeur de thèse

Remerciements

On exprime sans doute trop rarement sa gratitude, c'est donc avec plaisir et une émotion certaine que je me plie à l'exercice qui va suivre, celui des remerciements de thèse. J'ai eu de nombreux bienfaiteur.e.s pendant cette thèse et je leur dois entièrement la réussite de ces trois dernières années.

J'adresse d'abord ma reconnaissance à mon directeur de thèse, Fabien Alet. Je le remercie d'avoir été si impliqué, autant prêt à debugger mon code qu'à relire (très) attentivement ce manuscrit, d'avoir été compréhensif lorsque j'avais des moments de doute quand la thèse bloquait ou que j'avais des soucis personnels, de m'avoir encadré dans des projets variés et intéressants, et de m'avoir fait confiance lorsque je présentais nos travaux ou au moment de la rédaction d'articles. Il m'a aussi donné la chance de pouvoir assister à plusieurs workshops en Allemagne et au Canada (je regrette seulement les 5 tonnes de CO_2 que j'ai émises pour y participer..). Remerciement particulier pour cette école d'été d'un mois aux Houches qui a été une occasion unique de rencontrer les experts mondiaux de notre discipline dans un cadre de rêve, au pied du mont Blanc.

Je suis très heureux d'avoir pu rencontrer Evert van Nieuwenburg, d'abord à un workshop à Dresde puis au Perimeter Insitute à Waterloo. Fabien a bien fait de me pousser à aller travailler avec lui au Danemark. Ce fut un excellent séjour d'un mois et demi à Copenhague dans son équipe. J'ai eu plaisir à collaborer avec lui sur un projet original qui m'a plu au delà de mes attentes. Je garderai un très bon souvenir de l'institut Niels Bohr et de ses doctorants, moment que j'ai dû malheureusement écourter pour cause de pandémie mondiale.

Je remercie ensuite les rapporteurs Nicolas Regnault et Simon Trebst pour leur lecture attentive de ce manuscrit, j'ai trouvé leurs commentaires justes et instructifs. Je remercie également les examinateurs Eliska Greplova, David Guéry-Odelin et Evert van Nieuwenburg pour leur participation à ma soutenance ainsi qu'à la discussion que nous avons eu lors de mon oral. Je suis heureux d'avoir pu terminer ma thèse de cette façon.

Je remercie Malika et de Sandrine pour leur sympathie et leur disponibilité, ainsi que les membres du LPT, en particulier du groupe FFC, dont la présence un peu intimidante au début s'est révélée agréable et chaleureuse au fil du temps.

Je remercie très vivement la fondation CFM sans qui tout ceci n'aurait pas eu lieu. En particulier, le budget destiné à la mobilité qui a été englouti dans mon Airbnb à Copenhague (!).

Je remercie sincèrement mes collègues et ami.e.s du "bureau des doctorants" : les deux Benjamin, Julie, Jordan, Théo et Yassir. Grâce à vous, ce bureau était un endroit où je me sentais bien. Mention spéciale à mes camarades de la promotion thèse 2017-2020, Julie et Jordan, qui ont été les meilleurs (!), j'ai aimé nos discussions parfois légères, parfois profondes, toujours agréables. Je pense aux doctorants et post-doctorants de tous les étages avec qui j'ai partagé des repas, pauses cafés et plus aussi : les trois Maxime, le docte Olivier, Julie du LCAR, Julien, Nathan, Faedi, Ulysse, Pranay et Juraj. Pensée également à mes étudiants plus ou moins turbulents et/ou intéressés, j'ai apprécié mes différentes expériences d'enseignement qui me changeaient les idées quand la recherche n'avançait pas.

Je remercie enfin toutes les personnes qui m'ont accompagnées au quotidien à Toulouse, leur

présence a été importante pour moi : Lisa ma première coloc, Vincent du labo puis plus du labo puis en coloc aussi, Gabriel meilleur compagnon d'escalade et de sorties techno, Mariane avec qui j'ai découvert plein de choses et fait de beaux voyages, Agnès dont j'apprécie beaucoup la compagnie musicale ou sportive, Nicolas et Julia dont la présence bienveillante et joyeuse a transformé les difficultés de la dernière année (entre confinements et rédaction) en souvenirs *richtig* agréables, et enfin Marion qui apporte énormément de gentillesse et de douceur à ma vie.

Je n'oublie pas mes vieux amis qui comptent pour moi : Arsène, Lamia, Jules et Augustin. Et enfin, mes tantes et oncles, cousins, cousines, mes grands-parents, en particulier à ma Mouli qui est partie paisiblement l'été dernier. Tout ceci ne serait pas possible sans l'amour et le soutien constants de mes adelphes Zoé et Robin et de mes parents Hervé et Agnès, merci du fond du cœur.

Contents

Chapter 1

Introduction

1.1 Machine learning

For centuries, humanity has been fascinated by the possibility of non-human intelligence. An illustrative example of this, among so many others, is the story of the mechanical Turk. In the late 18th century, Hungarian inventor Wolgang von Kempelen constructed a chess-playing automaton that was able to defeat many challengers for nearly 80 years around Europe and America. The Turk was in fact a hoax, a mechanical illusion that allowed a skilled human player to hide inside to operate the machine (see Fig. 1.1).

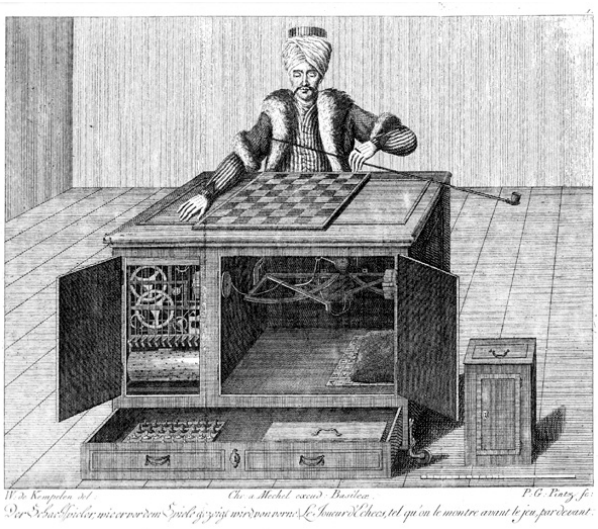


Figure 1.1: A copper engraving showing the Turk, the open cabinets where the human operator could hide and working parts, by [Gottlieb von Windisch's 1784].

Two centuries after, in 1997, the computer program Deep Blue was the first chess-playing machine able to defeat a human, none other than Garry Kasparov the world champion at the time, with a score of 3 wins, 2 losses and 1 draw. The core of this chess program was based on finely handcrafted evaluation functions that incorporated chess knowledge acquired by grandmasters from two centuries of game playing. In that sense, perhaps put provocatively, Deep Blue was just a modern version of the mechanical Turk, devising a chess machine based on human knowledge, though in a rightfully more sophisticated way. Twenty years after, the Deepmind team set another milestone in the field by devising AlphaGo [Silver *et al.* 2016], a program that was able to beat human champion Lee Sedol in the even more challenging game of Go. Shortly after, DeepMind introduced a new algorithm [Silver *et al.* 2017] achieving higher performance by learning the game only with self-play, that is, only input with the rules of Go, it could improve by

playing against itself. They later extended their result to the game of chess [Silver *et al.* 2018] demonstrating that, starting from random play, AlphaZero could reach superhuman level in chess after only 4 hours of self-play! The great superiority of machine learning (ML) for Go- or chess-playing clearly exemplifies its potential and explains its recent popularity as a new programming paradigm.

Although these superhuman-performing algorithms are impressive in their own right, one can argue that their capabilities largely rely on the greater computational power and memory capacity that computers have over humans. In other words, although being intellectually difficult for humans, chess and Go are straightforward to handle for computers as their rules and the game mechanics can be easily implemented. Therefore, one can think of a true challenge for artificial intelligence (AI) as one that involves tasks that are easy for people to perform but hard to describe formally: where intuition and automatisms play a big role.

Researchers took up this challenge and succeeded in developing machine learning algorithms that are now able to routinely solve complex real-world tasks such as detecting objects in an image, recognizing spoken words or translating sentences into another language. Even though the principles and methods have been known for almost 40 years, the field has greatly expanded in the last decade and is now getting attention across its frontiers, being applied in many domains like medicine, weather-forecast and even lately physics. The reason for this golden age stems from a favorable combination of factors. First, the birth of the Internet and social media has generated an enormous amount of data (40 trillion gigabytes as of today [Reinsel *et al.* 2018]) which quickly called for efficient analysis methods. Second, major theoretical breakthroughs [Hinton *et al.* 2006, Krizhevsky *et al.* 2017a] led to the birth of deep learning and triggered the biggest success in AI [LeCun *et al.* 2015]. Finally, the computational power of CPUs has been growing constantly over the years and the intensive use of dedicated hardware such as Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs) [Jouppi *et al.* 2017] has only helped this progress.

Deep learning

A learning algorithm is a program that is able to extract knowledge from experience. In other words its performance can be improved upon being exposed to more data. As for humans, AI systems have been designed to understand the world in terms of concepts. By being able to relate and compose concepts with each other, knowledge of increasing-complexity can build up. If we draw a graph showing the hierarchy of these concepts assembled on top of each other, the graph would be **deep**, i.e. containing many *layers*. The above general conceptualization is embodied by deep neural networks which in short is a function built as a composition of simple unit functions. We leave the details to chapter 2 for now.

For a human, the process of recognizing a cat in an image can be decomposed into the detection of low-level features such as color palette, texture of hairs, small patterns like ears, or feet, then higher-level characteristics such as body shape, bigger scale color motifs converging together to the concept of cat. Before deep learning methods were employed, say for a cat recognition algorithm, programmers would manually implement feature detectors and process the image through them before eventually providing a shallow algorithm with this human-processed data. The key advantage of deep learning lies in its ability to *automatically* find a suitable representation of the data, possibly containing high-level abstract conceptualization of

the input data, and effectively solve complex tasks.

Limits of machine learning

Although very efficient in many cases, machine learning techniques suffer from different types of limitations. Since this technology is heavily data-dependent, it is natural to expect that the quality of the data is directly reflected in the performance of these algorithms. There are many situations where collecting data is costly, data can also be corrupted or sparse, thus a machine learning practitioner should devote a substantial part of its time to data normalization. An other criticism that can be made is that the decision function of deep learning algorithms is notoriously known to be opaque. In other words, it is almost always impossible for a human to understand which features of the input dictate the final output of the algorithm. Interpretability is one of the biggest challenges of today's machine learning and the lack of notable progress in this direction actually puts a brake on its real-life applications [Goodman & Flaxman 2017].

On a different note, there is no established understanding *why* these algorithms work so well. Some even compare machine learning to steam engines [Hartnett 2019]. At first, engineers would improve them based on intuitions extracted from direct experimentation, until the theory of thermodynamics was developed and eventually gave birth to an industrial revolution. The current situation in machine learning is similar since most architectures and training tricks have been devised from empirical intuition. One example among many of the deep learning puzzles is the fact that deep networks that have more adjustable parameters than actual number of training samples still have good generalization properties [Zhang *et al.* 2017]. The field certainly calls for more systematic and theory-based approaches [Carleo *et al.* 2019b].

1.2 Numerical methods for condensed matter physics

Condensed matter theory is concerned with the study of mathematical models that are effective representations of real materials. These toy models are conceived to be rich enough to capture the essential traits of physical systems, while being simple enough to remain analytically (or numerically) tractable, the latter being achieved by discarding all the irrelevant features of the system at study in the modelling. New intriguing phenomena are being observed in experiments on a daily basis and some of them still resist a complete theoretical understanding. Long-standing problems in quantum many-body physics include high-temperature superconductivity [Bednorz & Müller 1986], the fractional quantum Hall effect [Tsui *et al.* 1982, Laughlin 1983] or lately superconductivity appearing upon twisting two sheets of graphene stacked on top of each other [Cao *et al.* 2018].

A relatively new branch of physics is devoted to the study of these complex systems using computational resources. Half-way between lab experiments and pen-and-paper calculations, computational physics deals with the in silico simulation of mathematical models. A convenient simplification for numerics is to consider effective lattice models where the degrees of freedom lie in a discretized space (which is justified in condensed matter as atoms organize periodically in space forming crystals). Arguably one of the most important example of such model is the Hubbard model [Hubbard & Flowers 1963], where electrons can hop on the vertices of a lattice and interact with each other via on-site repulsion which emulates the underlying Coulomb

repulsion:

$$\mathcal{H} = \underbrace{-t \sum_{\langle i,j\rangle,\sigma} c^{\dagger}_{i,\sigma} c_{j,\sigma} + \text{h.c.}}_{\text{electron hopping}} + \underbrace{U \sum_{i} n_{i,\uparrow} n_{i,\downarrow}}_{\text{Coulomb repulsion}} \tag{1.1}$$

where $\langle \cdots \rangle$ indicates neighboring sites on a given lattice in d spatial dimensions. The fermionic operator $c^{\dagger}_{j,\sigma}$ ($c_{j,\sigma}$) creates (destroys) one electron with spin σ (up $\uparrow$ or down $\downarrow$) on an orbital residing on site j.

The Hubbard model provides an exceptional platform for phase exploration in fermionic (Fermi-Hubbard model) as well as bosonic (Bose-Hubbard model) systems hosting several interesting phases of matter including metals, Mott insulators or superconductivity. Despite its apparent simplicity, its phase diagram is not known in systems of dimension larger than 2 and exact numerics are plagued in most regimes by the so-called sign problem (see below). The model is also naturally realized in quantum simulators, in particular cold atom gases loaded on optical lattice. The limit of large interactions gives rise to the Heisenberg model which we will study in chapter 3 (though with the addition of disorder).

In general, exact solutions of quantum lattice models are rare beyond one dimension, and usually analytical developments have to ultimately rely on approximations or assumptions that cannot always be rigorously justified. As a result, computational studies can help circumventing these limitations. Among other methods, we present in the following three representative examples of techniques that are routinely used in solving the many-body problem.

Exact Diagonalization

The most direct way to study a quantum spin model is to construct its Hamiltonian matrix and diagonalize it. This gives access to the exact eigenstates and energy spectrum of any finite size system. However, due to the exponential growth of the Hilbert space with system size (the Hilbert space size of N spin-1/2 particles is 2^N), the Hamiltonian matrix also becomes large exponentially fast and in practice, the limit in computational time and memory capacity is rapidly reached for a few tens of spins. Nevertheless, the unbiased nature of this method is of great interest for testing new methods on small lattices against exact exact diagonalization (ED) results. In some cases, a finite-size study supplemented by an appropriate scaling theory and symmetry analysis is sufficient to extrapolate observables to the thermodynamic limit and obtain a phase diagram [Wietek *et al.* 2017].

One way to push the method further is to take advantage of the symmetries of the model. Given that the Hamiltonian and the symmetry operators commute with each other, the Hamiltonian can be written in a block-diagonal form where each small block corresponds to different symmetry sectors [Sandvik *et al.* 2010]. If the focus is on the low-energy properties of the system, it is possible to target and obtain the exact ground-state and first excited states by employing Krylov-space-based techniques such as the Lanczos method [Sandvik *et al.* 2010]. The computational effort is thereby reduced but still remains exponential. Let us also mention the family of spectral transform methods (such as shift-invert [Pietracaprina *et al.* 2018a]) which enables to obtain eigenstates in the middle of the spectrum.

Variational methods: *compressing*

One strategy to circumvent the exponential complexity of a quantum wave-function is to parametrize it by a tractable number of parameters. More precisely, for a quantum state $|\Psi\rangle = \sum_i c_i |i\rangle$, the number of amplitude coefficients c_i grows exponentially as the system size increases. The aim of variational approaches is to encode these amplitudes succinctly with a number of variational parameters that only grows polynomially with system size. By construction, this approach is biased by the specific wave-function parametrization, nevertheless reliable approximations of many strongly correlated states can be found. This approach was pioneered by Feynman in his study of polarons and superfluid helium [Feynman 1955].

Given a Hamiltonian H and its ground-state $|E_0\rangle$, how can one approximate $|E_0\rangle$ with a certain class of wave-functions, i.e. a specific parametrization? One way to achieve this is to use the so-called *variational principle* that provides an inequality valid for any variational state $|\Psi_{\text{var}}\rangle$, relating the exact ground state energy E_0 and the variational energy E_{var} of $|\Psi_{\text{var}}\rangle$ (proved in Sec. 4.2 of chapter 4):

$$E_0 \leq E_{\text{var}} = \frac{\langle \Psi_{\text{var}} | \mathcal{H} | \Psi_{\text{var}} \rangle}{\langle \Psi_{\text{var}} | \Psi_{\text{var}} \rangle} \tag{1.2}$$

As a result, any trial state $|\Psi_{\text{var}}\rangle$ gives an *upper bound* of the exact ground-state energy. More importantly, equation (1.2) allows to control the approximation made by the variational state by simply following its variational energy E_{var}. Indeed, without knowing E_0, the lower the variational energy E_{var} is, the closer it is expected to be to the actual ground-state energy, and $|\Psi_{\text{var}}\rangle$ is expected to approximate the true ground-state wavefunction. Optimizing the variational parameters can be done with Monte Carlo simulations as will be explained in chapter 4.

One family of states that drew attention in the past decade and was able to achieve state-of-the-art descriptions of many one-dimensional and some two-dimensional systems, is matrix product states (MPS) [Verstraete *et al.* 2008]. Since Hastings foundational result [Hastings 2007], it is known that gapped one-dimensional quantum systems satisfy the so-called area law for the entanglement entropy. A direct consequence of this is the possibility to approximate *efficiently* the ground-states of such systems with MPS [Verstraete *et al.* 2008]. This means that the number of MPS parameters (the bond dimension) required to reach a given accuracy only grows *polynomially* with system size, thus completely removing the original exponential complexity of the problem. Unfortunately, MPS become inoperative whenever the amount of entanglement in the targeted state is too large, for example when it obeys a volume-law entanglement entropy. This can occur for example when the focus is not on the low-energy properties of the system but on highly excited states usually more entangled. This latter scenario arises in the study of closed quantum systems which will be discussed in chapter 3.

Quantum Monte Carlo: *sampling*

Another strategy to deal with the exponential complexity of quantum many-body systems is to statistically sample a quantum state with quantum Monte Carlo (QMC). Here the most relevant components of a wave-function or of a path integral are selected through *importance sampling* and this allows us to efficiently obtain accurate observable estimations with a number of samples

that grows polynomially with system size. Contrary to the previous approach, in the best-case scenario, QMC is unbiased.

In general, one can distinguish QMC methods that target the ground-state and those that specialize to finite-temperature properties. For the latter, the core idea of the method is to rephrase the partition function of a d-dimensional quantum system as the partition function of a $d+1$-dimensional classical system, i.e. $Z = \mathrm{Tr}\left(e^{-\beta\mathcal{H}}\right)$ is recast as a sum of statistical weights over classical configurations, $Z = \sum_C W_C$. When the weights W_C are positive, W_C/Z can be interpreted as a probability distribution and the usual Monte Carlo machinery can be applied straightforwardly with its efficiency as well as its pitfalls. Near a phase transition for instance, the autocorrelation time between samples can increase so much that the computational effort needed to get a sufficient amount of samples is prohibitive. The existence of efficient Monte Carlo updates is often required as we will see in chapter 4 and in some cases, efficient schemes were found to even cancel this phenomenon [Syljuåsen & Sandvik 2002, Alet & Sørensen 2003]. This latter fortunate situation arises for non-frustrated spin and bosonic systems and QMC is then the method of choice for accessing *exact* equilibrium properties at both zero and finite-temperature and any dimensions.

However, in many interesting cases such as frustrated or fermionic systems, the weights W_C turn to be negative or even complex, which invalidates the usual Monte Carlo interpretation of W_C/Z as a probability distribution, giving rise to the infamous *sign problem*. Let us first observe two simple facts, (i) having a Hamiltonian with only negative elements is sufficient to avoid the sign problem since the partition $Z = \mathrm{Tr}\left(e^{-\beta\mathcal{H}}\right)$ will directly be a sum of real positive terms, (ii) Z is invariant with respect to unitary transformations ($\mathcal{H} \to \mathcal{U}\mathcal{H}\mathcal{U}^\dagger$). As a consequence, if the Hamiltonian is expressed in its eigenbasis, its elements can all be made negative (by a global irrelevant energy shift) and the sign problem is solved... at the exponential cost of diagonalizing the Hamiltonian! All in all, the sign problem is purely a problem of representation and in some specific instances, it was in fact possible to solve it by ingeneously designing appropriate unitary transformations [Chandrasekharan & Wiese 1999, Chandrasekharan & Li 2012, Li *et al.* 2015, Alet *et al.* 2016].

In summary, at present, there are no exact methods that allow to probe eigenstates of general many-body Hamiltonians with a computational effort that scales polynomially with the system size. Although provably hard when the sign problem manifests [Troyer & Wiese 2005], the many-body problem has been driving the efforts of the computational quantum many-body physics community for decades and the future developments of the field might well be directed towards analog quantum simulators [Friedenauer *et al.* 2008, Bloch *et al.* 2012], quantum computers [Arute *et al.* 2019] and, as we will investigate in this thesis, coupling with machine learning techniques.

1.3 Machine learning and condensed matter physics

Physics does not escape from the general late trend of big data. As greater computational power and memory capacity are more accessible, datasets become larger and data samples of higher quality, for example in high-energy physics with data collected from the LHC or big telescopes. Physics problems also share structural similarities with ML tasks. For example, one can notice the parallel between the exponential complexity of an interacting many-body quantum system and the *curse of dimensionality* which affects machine learning and states that the training

dataset size should scale exponentially with the dimensionality of the input samples, as shown in Fig. 1.2.

Figure 1.2: As the dimensionality of the input data is increased (from left to right), the number of configurations grows exponentially. This means that eventually the size of the training dataset should scale with input dimension to probe all state space regions. Figure from [Goodfellow *et al.* 2015].

Beyond high-dimensionality, many-body systems and data in machine learning can be characterized by correlations and symmetries. For example, it was showed that pixel-to-pixel [Ruderman 1994] or word-to-word [Ebeling & Pöschel 1994] correlations are power-law decaying, which is reminiscent of the power-law spatial correlations of critical systems in many-body systems. In the same spirit, symmetries often help the understanding of quantum systems and can reduce the computational complexity of certain learning tasks.

Well known in statistical learning theory and used early on in the large datasets of particle physics experiments, machine learning became a new phenomenon in condensed matter physics only recently and started being applied to a large variety of problems such as the determination of atomization energies of molecules [Rupp *et al.* 2012, Pilania *et al.* 2013] or the acceleration of Monte Carlo simulations [Huang & Wang 2017, Liu *et al.* 2017a, Xu *et al.* 2017]. The two fields have already begun to cross-fertilize as is shown by the use of tensor networks for image classification [Stoudenmire & Schwab 2016] or the tentative interpretation of neural networks with renormalization group concepts [Mehta & Schwab 2014]. Numerous contributions also concerned experiments as the ML-assistance of quantum state tomography [Torlai *et al.* 2018], state preparation [Bukov *et al.* 2018] or parameter estimation [Greplova *et al.* 2017] proved very promising. In the following paragraphs, we introduce three other fields of application of ML to condensed matter physics, which will be the respective topics of the next chapters of this manuscript.

Machine learning phases of matter (Chapter 3) One of the first application of machine learning in condensed matter physics treated the problem of classification of phases of matter. Traditionally, the understanding of phases and the mapping of phase diagram is obtained after identification of an order parameter that is a physical observable that takes different values in the phases allowing for their clear distinction. Although this diagnostic is easy to find in simple models like the Ising model, in the vast majority of cases, it is not always clear what is the order parameter and even its existence may not be guaranteed. A fresh look on this problem came from a series of papers [Wang 2016, Carrasquilla & Melko 2017, Wetzel 2017, van Nieuwenburg *et al.* 2017, Broecker *et al.* 2017a] in which a machine algorithm was set up to achieve the phase classification task autonomously. This approach applies equally well to

synthetic and experimental data (see Fig. 1.3) and paves the way to exciting possibilities such as the discovery of new order parameters.

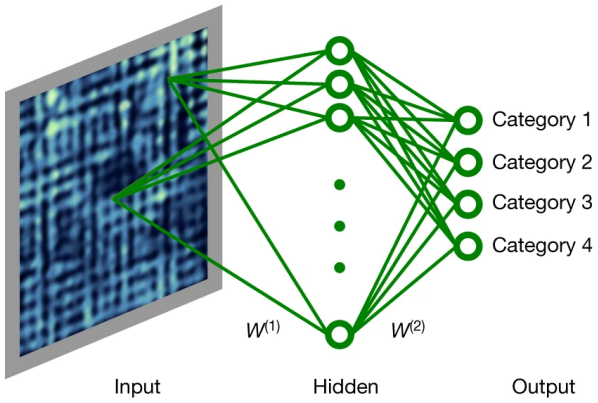


Figure 1.3: Example of a machine learning approach to the classification of phases: an experimental image (here scanning tunneling microscopy of high-temperature superconductors) is sent to a neural network which classifies it as belonging to one of the preset categories (here different spatial modulations). Figure adapted from [Zhang & Kim 2017].

Neural-network quantum states (Chapter 4) An entire research field opened up following the seminal work of [Carleo & Troyer 2017] who propose to parametrize the amplitudes of a quantum many-body wave function with a neural network. So-called neural-network quantum states (NQS) write $|\Psi\rangle = \sum_i f(i)|i\rangle$, where the basis states $|i\rangle$ are chosen to be indexed by quantum numbers e.g. $|\uparrow\downarrow\cdots\uparrow\rangle$ for a system of spin-1/2 particles. The neural network f then simply takes the vector of quantum numbers indexing $|i\rangle$ and returns the corresponding amplitude $f(i)$. [Carleo & Troyer 2017] showed that NQS can be used as variational approximations of quantum many-body ground states, achieving excellent performance for 1d and 2d spin models. The complexity of the representation directly follows from the architecture of the underlying network f, which provides great flexibility and expressive power to this ansatz as increasing the width or depth of f is likely to improve the variational approximation.

Self-learning decoders for quantum error correction (Chapter 5) Quantum error correction deals with the protection of quantum information. Topological quantum codes have recently emerged as promising candidates for the implementation of qubits [Kitaev 2003]. By encoding one logical qubit in the topological properties of a larger system (that can be composed of many physical spins), the effect of decoherence (individual spin flip for instance) can be actively corrected. This task can be assisted by machine learning techniques in many ways [Torlai & Melko 2017, Varsamopoulos *et al.* 2017, Sweke *et al.* 2018]. By formulating the correction process as a game, it is for instance possible to use the same kind of methods that led to the recent breakthrough in the game of Go [Silver *et al.* 2016].

1.4 Organization of the manuscript

This thesis is organized in four chapters.

- Chapter 2 is an introduction to the methods of machine learning and is written to provide the minimal knowledge needed for the understanding of the rest of the manuscript.
- Chapter 3 focuses on the application of ML to the study of phases of matter. First, the methods allowing for an automatic detection of phase transitions are described. Then I present the results of these approaches on the many-body localization problem [Théveniaut & Alet 2019, Théveniaut *et al.* 2020].
- Chapter 4 describes how neural-network quantum states can be used in variational Monte Carlo and in projection methods. I will show how this approach can help study the ground-state properties of a constrained two-dimensional bosonic model with competing ring-exchanges.
- Chapter 5 presents preliminary work on the application of an evolutionary algorithm to quantum error correction.

Chapter 2

Methods of machine learning

Contents

Before describing the methods of machine learning, we need to define what means *learning* for a machine. Answering this question inevitably leads us to build a general theory of learning, which also concerns human learning. Indeed, if one is able to formalize the learning experience as a set of fundamental rules and processes, then its simulation in a computer would become possible. The foundations of reasoning have been discussed first at the time of Ancient Greece when philosophers introduced seminal ideas in logic distinguishing different types of reasoning like deduction (where general rules allow to draw conclusions on specific examples) or induction (where specific examples allow to infer a general rule applicable to new examples).

In most cases, learning is concerned with a **task** that an agent, a human or a machine, wants to achieve. At the beginning, the agent does not know the appropriate actions to make progress but if it is exposed several times to the same task, one can expect that it will slightly adjust its actions for the next trial based on success or failures of the previous attempts: this is the general trial-and-error approach. A more formal definition of this was given by Tom Mitchell [Mitchell 1997]: "*A computer program is said to* ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*"

The aim of this chapter is to give a condensed introduction on machine learning (ML) with a focus on the learning algorithms that I have been using throughout my PhD. First section deals with presenting the core components of any machine learning algorithm. In second section, we will solve a concrete simple example of a supervised ML task that will highlight many important practical notions. In the third section, we will introduce an extremely popular learning model called neural networks. Fourth section will present two dimensionality reduction techniques. Finally, the fifth section will give a short introduction to the methods of reinforcement learning.

2.1 Machine learning basics

One can think of two categories of programs: on the one hand, hard-coded algorithms written by human experts, on the other hand, adjustable algorithms that can change after being exposed to examples of the task. In the latter scenario where human intervention is greatly reduced one needs to set up an appropriate environment in which this type of algorithm can evolve in an autonomous manner, or put differently, *learn*.

The goal of this section is to introduce the four essential components of any machine learning method: a dataset, a learning model, a cost function and an optimization procedure. Afterwards, we will give examples of day-to-day applications of these techniques.

2.1.1 Dataset

In the most general form, a dataset $\mathcal{D}$ is a collection of a finite number N of examples, $\mathcal{D} = \{(\mathbf{x}_i)\}_{i=1,\dots,N}$, where each example or data point is a real vector of possibly large dimension. A general assumption made in ML is that the dataset is obtained from sampling a data probability distribution $p_{\text{data}}(\mathbf{x})$, that is generally unknown. It is also commonly assumed that the data samples are drawn independently and identically distributed from p_{data}. An example of dataset is shown in Fig. 2.1.

In some cases, each data sample $\mathbf{x}_i$ is paired up with a label or a target value y_i which is usually given by a human expert. For example, for the MNIST dataset, each image is provided with a number from 0 to 9 that simply tells which digit is shown in the image. Another popular dataset used for benchmarks is ImageNet which contains 14 million of images distributed over 20000 categories [Russakovsky *et al.* 2015]. When a label is provided with each data sample, learning is said to be **supervised** in the sense that a human indicates to the algorithm which output y_i is expected to be returned from an input $\mathbf{x}_i$. When no label is given, learning is said to be **unsupervised**. We will give examples of both cases in Sec. 2.1.5.

Figure 2.1: Subset of images originating from the MNIST dataset [LeCun *et al.* 1998]. The dataset consists of 60000 images of handwritten digits of size 28×28. Here each data point $\mathbf{x}_i$ is an image encoded as a real vector of 784 dimensions filled with greyscale pixel values.

2.1.2 Model

Depending on the task, a model can be viewed as a deterministic function or a probabilistic estimator that depend on internal parameters $\boldsymbol{\theta}$. For instance, we will consider models in Sec. 2.2 that are polynomials parametrized by their coefficients. In Sec. 2.3, we will introduce another family of functions called neural networks. An important notion associated to a model is its **representational capacity** which can be roughly estimated as the number of parameters of the model (e.g. the degree of the polynomial). Most considered models have good representational properties such as being able to approximate arbitrarily well continuous functions, which comes in the form of **universal approximation theorems**.

At first sight, supervised learning tasks may seem very similar to interpolation. Indeed, given a training dataset $\mathcal{D}_{\text{training}} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$, the learning algorithm is asked to find a model $f_{\boldsymbol{\theta}}$ that approximates well the mapping $\mathbf{x}_i \xrightarrow{f_{\text{data}}} y_i$ (i.e. $f_{\text{data}}(\mathbf{x}_i) = y_i$), that is such that $f_{\boldsymbol{\theta}} \approx f_{\text{data}}$ in a sense that we will clarify later. The difference with interpolation techniques comes from the fact that every ML model is expected to perform well not only on the training dataset but also on new samples: it should be able to **generalize** well. Alternatively, machine learning algorithms can be considered as probabilistic predictors. Put in the probabilistic language, a model $p_{\boldsymbol{\theta}}$ is asked to approximate well the data distribution, either the full distribution $p_{\text{data}}(\mathbf{x})$ in unsupervised learning, or the conditional probability distribution $p_{\text{data}}(y|\mathbf{x})$ in supervised learning.

One can now ask the following question: is there a best model? A firm negative answer was given in 1997 by Wolpert and Macready [Wolpert & Macready 1997] that proved the famous **no free lunch theorem**. This theorem states that, averaged over all possible data-generating distributions, every classification algorithm has the same accuracy when classifying unobserved data. In other words, no machine learning algorithm is universally better than any other, and in particular random guessing. Fortunately, these results hold only when we average over all possible data-generating distributions, therefore it does not forbid the existence of algorithms that perform better than others for specific data distributions. As we will see in Sec. 2.3, ML researchers have exploited the characteristics of real-world data to design efficient learning models.

2.1.3 Cost function

Let us now clarify in which sense we consider that a model is close to the true underlying data-generating model (i.e. $f_{\boldsymbol{\theta}} \approx f_{\text{data}}$). For supervised learning, a **cost function** is designed to measure the distance between the model predictions $f_{\boldsymbol{\theta}}(\mathbf{x}_i)$ and the target y_i, training will then consist in minimizing it with respect to the model parameters $\boldsymbol{\theta}$. For continuous labels, one usually uses the mean-squared error (MSE):

$$\text{MSE}(f_{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{i=1}^{N} ||y_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)||^2 \tag{2.1}$$

For categorical labels (i.e. taking values in $\{0, 1\}$), the most commonly used loss function is the cross-entropy, defined as follows:

$$H(f_{\boldsymbol{\theta}}) = \sum_{i=1}^{N} y_i \log(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \tag{2.2}$$

The cost functions in Eqs. (2.1) and (2.2) define different performance metrics and the best model given these metrics has parameters $\theta_{\text{opt}} = \underset{\theta}{\text{argmin}}\ \mathcal{L}(\theta)$, with $\mathcal{L}$ being the chosen cost function.

As evoked earlier, the final goal of a learning algorithm is to generalize well on unseen data, therefore the evaluation of a model should not only rely on its training performance, one should also quantify its generalization power. To do so, the dataset is first partitioned between a **training set** that the model has access to during training time and a **test set** not seen during training which allows to evaluate the predictions of the model on new unobserved samples. The generalization performance can then be assessed by computing the MSE on the test set for continuous labels, or the accuracy on the test set (the proportion of correct predictions) for categorical labels. Generalization performance will be discussed in more details in Sec. 2.2.2.

2.1.4 Optimization

It is possible for certain cost functions to obtain the optimal parameters $\boldsymbol{\theta}$ analytically i.e. in closed form. This usually happens for linear methods, however when non-linearity arises, it is most often necessary to set up an iterative numerical optimization to minimize the cost function. One of the simplest techniques is the algorithm of gradient descent. This technique is based on the observation that in the neighborhood of a point $\boldsymbol{\theta}$, a function $f(\boldsymbol{\theta})$ decreases fastest if one goes from $\boldsymbol{\theta}$ in the direction of the negative gradient of f. This leads to the following iterative algorithm:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \gamma \vec{\nabla} f(\boldsymbol{\theta}_n) \tag{2.3}$$

For $\gamma > 0$ small enough, we have $f(\boldsymbol{\theta}_n) \geq f(\boldsymbol{\theta}_{n+1})$. γ is a free parameter of the learning algorithm called the **learning rate** in the field of ML, it is one of the most important **hyperparameters**. The gradient descent method generates a sequence of points in parameter space that will always converge to a **local minimum**. For convex functions, all local minima are also global minima, unfortunately in ML, we are faced with hard non-convex optimization problems where gradient descent will most likely get stuck in local minima. We will discuss later the implication of this in the context of neural networks (see Sec. 2.3). As shown in Fig. 2.2, there is a tradeoff to find

between slow convergence at small γ and oscillations and divergence at large γ where essentially the next proposed iteration always overshoot the local minimum. As we will see in Sec. 2.2.2.3 or Sec. 2.3.4, hyperparameter selection is a crucial step in the training phase.

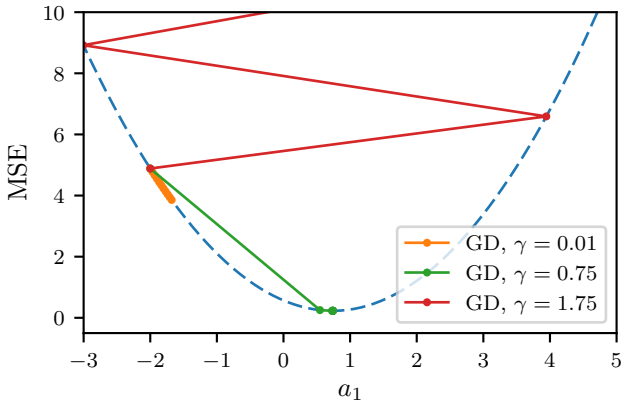


Figure 2.2: MSE as a function of a_1 for the training set shown in Fig. 2.3 with a linear model $f(x; a_1) = a_1 x$. The gradient descent is employed to find the minimum of the MSE with respect to a_1, the first 10 iterations are shown with an initial point at $a_1 = -2$ and three different learning rates γ.

2.1.5 Examples of machine learning problems

To get a sense of the type of problems machine learning is able to tackle, let us present some examples. The most common form of machine learning is *supervised learning*, here are some typical tasks:

- **Classification**: For object recognition or image classification, a machine learning algorithm is asked to detect the presence of objects in an image or assign a category to an image. There, the dataset is composed of pairs of images and labels. For example, to classify images of cats and dogs, the $\mathbf{x}_i$ are large vectors containing the pixel values (for example with RGB encoding) of the images and the y_i is 0 if $\mathbf{x}_i$ contains a cat or 1 if $\mathbf{x}_i$ contains a dog. In the case of object classification with multiple categories of objects, the targets $\mathbf{y}_i$ are usually encoded as one-hot vectors[1]. In this type of task, the learning algorithm should produce a function f taking value in $\mathbb{R}^d$ (where d is the dimension of a flattened image) and returns a value in $\{1, \dots, k\}$. The most famous example of this task is the image classification challenge based on the dataset "ImageNet" that stimulated research over the past decade and led to many important contributions to the field [Krizhevsky *et al.* 2017b, He *et al.* 2016].

- **Regression**: When the target/category y_i takes a continuum of values rather than a discrete one, the task lies in the category of regression problems. In Sec. 2.2, we will consider such a regression task in a simple scenario.

[1] If there are k categories and image $\mathbf{x}_i$ belongs to the kth category, then the components of $\mathbf{y}_i$ are zero except its kth component which is 1.

- **Image super-resolution**: An example of a less common task is image super-resolution whose task is to produce a higher resolution version of an image that is low-quality [Dong *et al.* 2014]. In this case, the output space is actually larger than the input space since there are more pixels in the output image.

Overshadowed by the success of supervised learning, *unsupervised learning* has been applied in many different contexts.

- **Clustering**: These tasks involve finding structures in the data like partitioning the dataset into groups of elements that share the same properties (see for instance the k-means algorithm [MacQueen 1967]).

- **Dimensionality reduction**: As we will see in Sec. 2.4, there exist methods like principal component analysis [Pearson 1901] (PCA) or autoencoders [Hinton & Salakhutdinov 2006] that aim at searching for a lower dimensional representational space that efficiently keeps the essential features of the data. These techniques are useful for noise reduction, data visualization or as an intermediate step that can facilitate other analyses.

- **Generative modelling**: In this type of task, the machine learning algorithm is asked to generate new examples that are similar to those in the training data. It is usually done with techniques similar in spirit with dimensionality reduction, where a simplified model distribution is found to approximate the true data-generating distribution. One striking application has been the generation of new art pieces after a generative adversarial network [Goodfellow *et al.* 2014] has been trained to approximate the probability distribution of portraits painted from the 14th to the 20th century [Christie's 2018].

2.2 A paradigmatic example: Polynomial regression

The following section aims at showing a simple application of machine learning, from training to evaluation. To keep the discussion simple, we will deal with an *ad hoc* supervised task, a polynomial regression applied to two-dimensional data. The discussion below is inspired by the one given in [Mehta *et al.* 2019], although we changed the dataset. Despite being trivial compared to most ML problems, this example allows to highlight many of the important concepts underlying ML methods.

2.2.1 Training of the model

Let us consider a dataset made of N pairs of real numbers, i.e. $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\dots,N}$, which are represented as points in a plane as shown in Fig. 2.3. The task is to find a function f that can accurately find the mapping $x_i \to y_i$. For demonstration purposes, the coordinates y_i have been generated from the equation $y_i = \cos(x_i) + \varepsilon_i$ where ε_i is an independent Gaussian noise.

Polynomial regression deals with learning models that are polynomials of arbitrary degree, which can write:

$$f(x; a_0, \dots, a_d) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d \tag{2.4}$$

where $\boldsymbol{\theta} = (a_0, \dots, a_d)$ are real coefficients and constitute the parameters of the model to be learned. The capacity of f is determined by the degree d of the polynomial which is related to

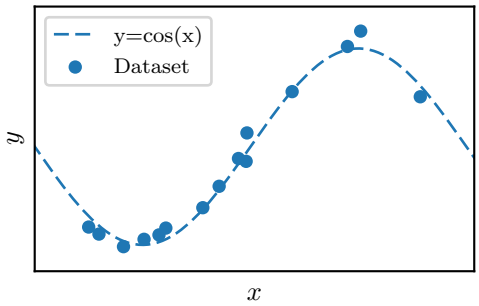


Figure 2.3: Dataset of points whose coordinates (x_i, y_i) are generated from the equation $y_i = \cos(x_i) + \varepsilon_i$ where ε_i is an independent Gaussian noise with zero mean and standard deviation $\sigma_\varepsilon = 0.1$.

the number of free parameters of the model. Increasing the degree of the polynomial in principle allows to represent more complex functions. An important property of polynomials is that they can approximate any continuous function defined on a closed interval.

Training of the parameters of the model is carried out by minimizing the mean-squared error (MSE) between the prediction of the model $f(x_i)$ and the true value y_i as shown in Eq. 2.5. Contrary to most ML tasks, the chosen model and cost function allow to fit the polynomial in closed form.

$$\mathrm{MSE}_{\mathrm{train}}(f; a_0 \dots a_N) = \frac{1}{N} \sum_{i=1}^{N} |f(x_i; a_0 \dots a_N) - y_i|^2 \tag{2.5}$$

Training results. Fig. 2.4 illustrates three situations that can occur as the complexity of the model f (d here) is varied. In all cases, a fixed-degree polynomial has been optimized to minimize the MSE on the same training set.

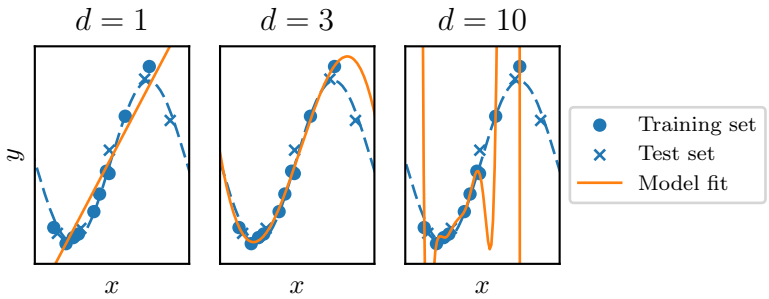


Figure 2.4: Three models of capacities increasing from left to right fitted on the same training set.

In the leftmost plot, the model is affine, it is clear that it is too simple to capture the curvature in the point locations, therefore we say it **underfits**. In the rightmost plot, a polynomial of degree 10 is used to fit the training data, it is capable of exactly passing through all the training points but it does not provide good predictions for new samples as the ones in the test set

indicated by crosses. In this case, the number of free parameters of the model actually exceeds the number of training points, as a consequence there is little chance of choosing a solution that generalizes well, there is **overfitting**. The appropriate capacity is found in the middle plot where the model fits well the training data and captures the right trend, leading to good generalization ability. A more detailed analysis of the generalization performance is given in the next sections.

2.2.2 Evaluation of the model

The generalization power of model f can be assessed by evaluating the MSE on the test set. We show next the effect of the model capacity, the size of the training dataset as well as of regularization on generalization.

2.2.2.1 Effect of model capacity

Fig. 2.5 summarizes the different error regimes depending on the capacity of the model. The training error is a monotonous decreasing function of the capacity, which is naturally expected since higher degree polynomials can fit more easily data. At sufficiently large capacity, the training error is even lower than the inherent noise of the data (the so-called Bayes error) which essentially means that the model has memorized the entire training dataset. The generalization error is always greater than the training error since the model is more likely to fail on examples it has not seen during training. The most notable feature of the figure is that the test error follows a U-shape because at low and high capacity the model underfits or overfits leading in both cases to bad generalization.

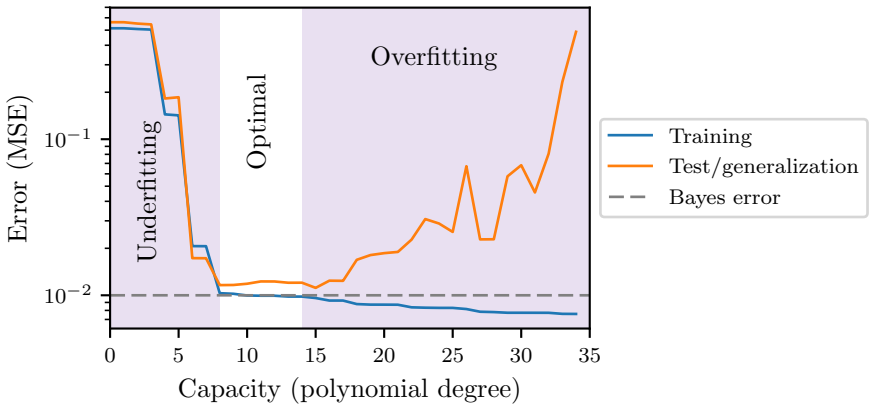


Figure 2.5: MSE with respect to training (blue) and test (orange) sets for different polynomial models plotted against model capacity for the dataset presented above. Underfitting and overfitting regimes arise whenever the capacity is respectively lower or greater than the complexity of the dataset. The Bayes error corresponds to the inherent noise of the dataset σ_ε^2 as defined in Fig. 2.3.

2.2.2.2 Effect of dataset size

Fig. 2.6 shows the effect of the size of the dataset on the generalization performance. Here, we pick three models of varying capacity and evaluate their training and generalization performances against augmenting the size of the training dataset.

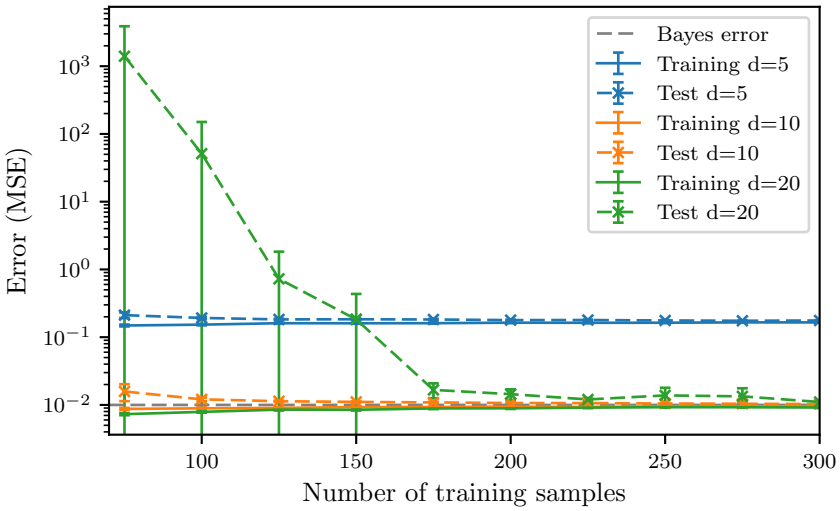


Figure 2.6: MSE on the training set (solid lines) and test set (dashed) plotted against the number of training samples for three polynomials of varying capacity. We ran independent trainings at each fixed dataset size, this means that every model will be trained and evaluated on 100 different datasets resampled from the underlying data distribution. Also, the test set size is set proportional to the training set size (being half as big). The error bars come from averaging over these 100 independent runs.

For all models, the generalization error always decreases when the training dataset is larger. This is what we naturally expect from learning since the more diverse samples the algorithm has experienced the easier it will be able to relate new samples to the ones it has seen. Although less visible in the figure, the training error slightly increases as there are more training samples. This can be explained by the difficulty for a fixed-capacity model to completely account for the complexity of a big and diverse set of data points. The low-capacity model (a 5th-degree polynomial) was shown to be in the underfitting regime in previous section. Fig. 2.6 confirms this since its error is the highest in the large dataset limit. This systematic error is the manifestation of its limited representational capacity that prevents it from fully capturing relevant features of the dataset, we say it is **biased**. It is notable that for small training dataset simple models perform better on average than large-capacity models. The large-capacity model (a 20th-degree polynomial), which was overfitting in the previous section, still displays poor generalization performance for small datasets but gets better as the number of training samples increases, reaching the same error rates as the $d = 10$ polynomial model. One can notice the important fluctuations (error bars) from one run to another in this case. This can be explained by the large capacity of this model that allows it to fit perfectly the small training datasets. We say this model has a high **variance**. Finally, the model of appropriate capacity (10th-degree polynomial) achieves the lowest training and test errors even for moderately big datasets.

2.2.2.3 Effect of regularization

As shown in Sec. 2.2.2.1, we can tune the representational capacity of polynomials by changing their degree and achieve the lowest training and test errors with moderate capacity. It is one central concern of machine learning to devise techniques that allow to improve the generalization ability without degrading the performance on the training set. Every technique that aims at achieving this is called **regularization**. This section is devoted to weight decay which is one of the simplest regularization techniques among many that we will discuss in Sec. 2.3.5. Weight decay consists in adding a term in the cost function that penalizes large parameter norms. This is implemented as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \text{MSE}(f_{\boldsymbol{\theta}}) + \alpha||\boldsymbol{\theta}||_2^2 \tag{2.6}$$

During the minimization of the cost function $\mathcal{L}(\boldsymbol{\theta})$, the optimization procedure should make a trade-off between minimizing the MSE on the training set and minimizing the L_2 norm of the coefficients of the polynomial. Fig. 2.7 shows that this technique allows to control the effective capacity of a model. In the leftmost curve, for large α, the constraint on the parameters of the model is too strong and leads to underfitting: the model is unable to capture the variations of the data. With no regularization, we recover the overfitting model that has too many parameters. For moderate regularization strength (middle plot), the large capacity of the model is balanced by the constraint on the weights and captures better the data structure.

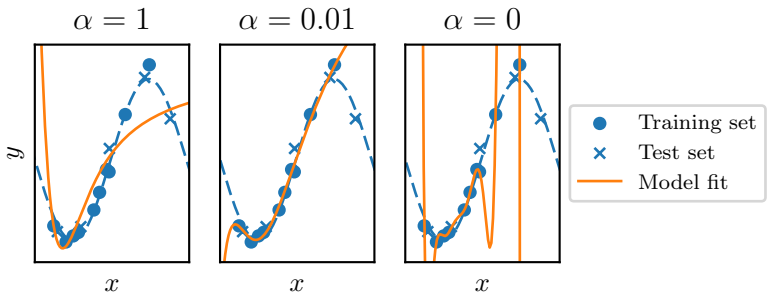


Figure 2.7: Results of training on the same training dataset with varying weight decay α. The model without regularization is a polynomial of degree 10 as the one in Fig. 2.4.

2.2.3 The bias-variance tradeoff

It is instructive to interpret the previous results in light of the concepts of bias and variance, two notions that sit at the heart of machine learning.

Bias refers to the ability of a model to describe a particular target function h. A model is a family of functions spanned by parameters $\boldsymbol{\theta}$, i.e. it can be written as $\mathcal{M}_f = \{x \mapsto f(x;\boldsymbol{\theta}); \boldsymbol{\theta} \in \mathbb{R}^d\}$. Fig. 2.8 shows the target function h as a red point and two models of different complexities: a complex model that spans a large area in the functional space (shaded green area) and a simple model (shaded grey area). If $h \in \mathcal{M}_f$, the model has no bias with respect to the solution, it can describe perfectly the target function. This is the case for the complex model which includes the target h, conversely, the simple model has a bias. Regarding the dataset studied

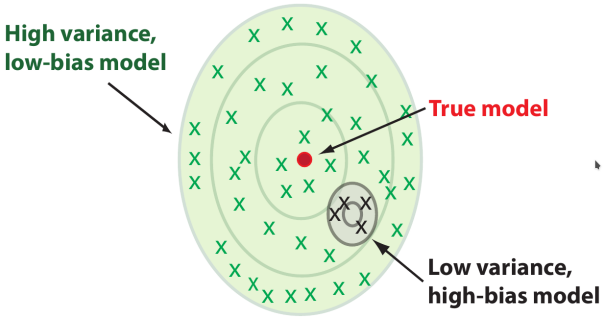


Figure 2.8: Schematic illustration of the bias-variance tradeoff. Figure from [Mehta *et al.* 2019].

previously, the polynomial models are biased since the function to be approximated is cos and $\cos(x) = \sum_{n=0}^{+\infty}(-1)^n \frac{x^{2n}}{(2n)!}$ which is a polynomial of infinite order.

The *variance* of a model refers to its sensitivity to fluctuations of the dataset. Indeed, a dataset can be seen as being generated from random sampling of a finite number samples from the data probability distribution. If the model is very sensitive to the specific samples it sees during training, it is said to have large variance. In fig. 2.8, the crosses indicate the different realizations of the two models after training on different datasets of the same size (drawn from the same distribution). As shown in the figure, the complex model (green) has high variance compared to the simple model (grey).

We refer to the very good discussion given in [Mehta *et al.* 2019] that poses in quantitative terms these concepts and derive the following relation on the generalization error:

$$E_{\text{generalization}} = \text{Bias}^2 + \text{Variance} + \text{Noise} \tag{2.7}$$

where the bias and variance terms are defined rigorously in [Mehta *et al.* 2019], the noise term is related to the inherent random nature of the dataset (ε in our previous test case).

We finish this section by giving guidelines to set up a ML task. It can be summarized as follows:

1. Collect and pre-process the data
2. Define the model and its architecture
3. Choose the cost function and optimizer
4. Train the model
5. Evaluate and study the model performance on the test data
6. Use the validation data to adjust the hyperparameters to optimize performance for the specific dataset

2.3 Neural networks

The first example of a neural network was introduced in the 1940s in the form of perceptrons [Rosenblatt 1957]. It was first thought as a computational model of the brain with the hope

that the dynamics of learning in these models would shed some light on how decision-making is achieved in the brain. Nowadays the first motivation was completely overshadowed by the amazing efficiency of these models at discriminative or generative tasks on data like image, sound or video. Moreover, the number of nodes in current artificial neural networks barely reaches the number of neurons of a frog which is roughly 6 orders of magnitude smaller than for the human brain [Goodfellow *et al.* 2016]. In previous section, polynomial regression was performed with models that depend linearly on their parameters (the function $(a_0, \dots, a_N) \mapsto f_{a_0,\dots,a_N}$ being linear). Neural networks can be seen as a natural non-linear generalization of linear models which are known to fail as soon as the data become more intricate.

The following sections will introduce a few of the most important neural network architectures currently used. We will introduce feed-forward neural networks (FFNNs) used for general purpose supervised learning, then convolutional neural networks (CNNs) that were designed specifically for image processing and finally restricted Boltzmann machines (RBMs) that are most often used in the context of generative modelling. At the end, we will discuss the specificities of training and regularization when learning models are neural networks.

As a disclaimer, we will not discuss support vector machines [Cortes & Vapnik 1995] or kernel methods [Guyon *et al.* 1993] in general, nor the very popular recurrent neural networks [Rumelhart *et al.* 1986] (RNNs) used for temporal or sequential data.

2.3.1 Feed-forward neural networks (FFNN)

Neural networks are composed of **units** or **neurons** with properties loosely resembling real neurons. In a nutshell, artificial neurons are simple functions that can take multiple input values and returns an output after some nonlinear operations. The perceptron neuron is defined as follows:

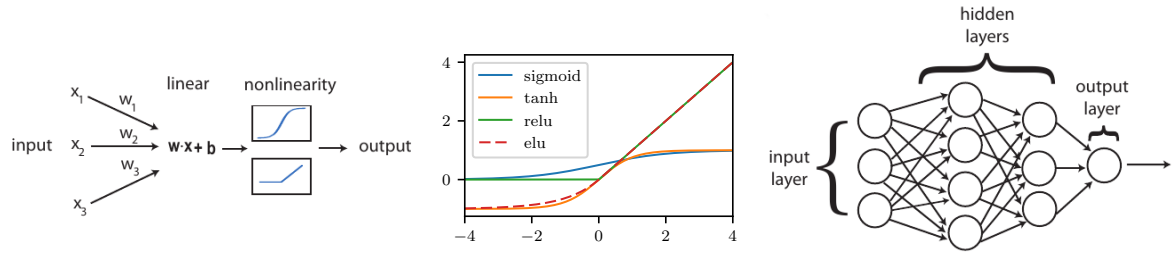
$$\text{Neuron}(\mathbf{x}) = H(\mathbf{W}^T\mathbf{x} + b) \tag{2.8}$$

where H is the Heaviside function defined as $H(x) = 1$ if $x > 0$ and $H(x) = 0$ otherwise, $\mathbf{W}$ and b are respectively a vector of weights and a scalar bias, and $\mathbf{x}$ a real vector of input values. Thus, neurons are small nonlinear functions that are the result of an affine transformation of the input followed by a nonlinear operation, carried out by a so-called **activation function** (here the Heaviside function H) as can be seen in Fig. 2.9(a). The most common functions are depicted in Fig. 2.9(b), in particular the rectified linear units ($\text{ReLU}(x) = \max(0, x)$) is currently the most popular one for deep architectures since it was showed that it allowed for faster training [Glorot *et al.* 2011].

Neural networks as their name suggests are built by connecting neurons with each other. Neurons are usually arranged in successive **layers** (see Fig. 2.9(c)). A layer is an array of neurons which processes simultaneously the incoming vector of values. Eq. 2.8 can be slightly changed to take the form:

$$\text{Layer}(\mathbf{x}) = H(\mathbf{W}^T\mathbf{x} + \mathbf{b}) \tag{2.9}$$

where in this case H is the Heaviside function applied element-wise on the incoming vector, $\mathbf{W}$ and $\mathbf{b}$ are respectively a weight matrix of size $N_o \times N_i$ and a bias vector of size N_o (N_i is the size of the input vector and N_o the number of neurons in this layer). As shown in Fig. 2.9(c), a layer of neurons located between the input and output layers is called **hidden**. In this figure, the network is **fully-connected** because each neuron from one layer is connected to all neurons



(a) (b) (c)

Figure 2.9: (a) Decomposition of a neuron, linear and nonlinear parts are exposed, from [Mehta *et al.* 2019]. (b) Most commonly used activation functions in deep learning applications. (c) Basic architecture of a feed-forward neural network, from [Mehta *et al.* 2019].

in the next layer. Finally, such neural-networks are called **feed-forward** because there are no recurrent connections in the network, the flow of computations in the neural network only goes forward, towards the output layer.

Deep learning. A neural network architecture that is obtained by stacking multiple layers of neurons on top of each other is called *deep*. The huge interest raised by deep neural networks stems from the fact that they are particularly suited to represent hierarchical concepts in which higher-level features are obtained by composing lower-level ones. This explains the success of these models when applied to datasets made of hierarchical data like texts, sounds or images. For example, detecting a cat in an image means that an algorithm has to somehow learn the concept of a cat, which is itself obtained as a composition of other concepts like the concept of a tail or triangular ears, which are again themselves composition of low-level concepts of shapes and textures. Fig. 2.10 shows how the concept of a person can be represented in a deep neural network: each layer transforms the input image by extracting low-level features, combining them in the next layer to detect increasingly abstract or bigger scale concepts, eventually being able to assign a label to the image.

The compositional nature of deep architectures allows to represent data in an exponentially compact form. One layer of n binary neurons (each neuron detecting a distinct feature in the data) can represent 2^n different concepts, and since the representations of two successive layers compose with one another, a neural network with L layers can in theory encode $(2^n)^L$ possible concepts. The exponential advantage is two-fold coming from the fact that representations are distributed among the neurons inside each layer and across layers. This also enables generalization to new combinations of the learned features beyond those seen during training. Another important point is the existence of a universal theorem stating that, given enough neurons, any continuous function can be arbitrarily approximated by a neural network [Cybenko 1989]. However, one of the main difficulties of the field is that in general, choosing the right architecture (number of nodes per layer, number of layer, which activation function,..) mostly resorts as of today on the empirical intuition developed by the ML practitioners and largely depends on the task and data at stake.

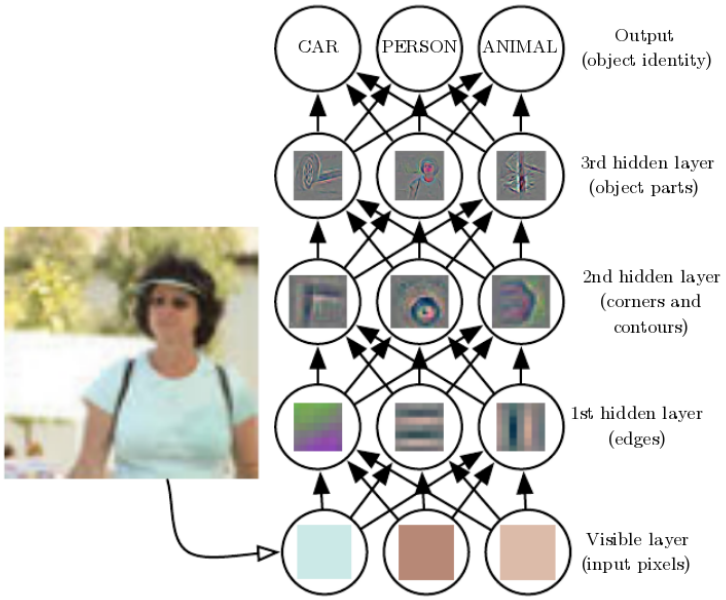


Figure 2.10: Schematic illustration of how deep neural networks build up by combining concepts of increasing complexity. The first layers will be able to detect low-level features such as color gradients or small motifs, while the next levels will combine these features making possible to detect higher scale components such as shapes, then entire objects until outputting the most abstract representation of the image being its label. Image from [Goodfellow *et al.* 2016].

2.3.2 Convolutional neural networks (CNN)

We have seen previously that the structure of neural networks are particularly suited to data that is structured in a hierarchical manner. Real-life images have the property to be translation and scale invariant (a translated or bigger dog on a image is still a dog) and respect locality (short-range pixel-pixel correlations [Ruderman 1994]) which are not properties implemented in the structure of fully-connected neural networks. Convolutional neural networks (CNNs) have been thought to fix this and has recently emerged as one of the most efficient architecture in image-based tasks [LeCun *et al.* 1989, LeCun *et al.* 2010, Krizhevsky *et al.* 2017b]. Fig. 2.11 shows the typical architecture of a CNN:

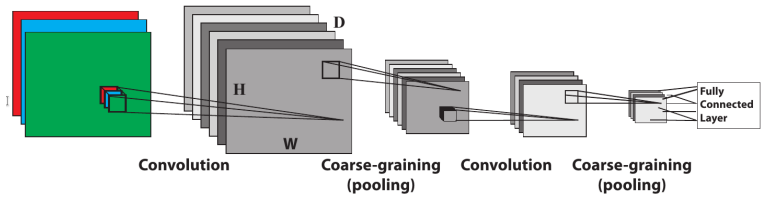


Figure 2.11: Architecture of a Convolutional Neural Network. Image from [Mehta *et al.* 2019].

Convolutions. The input image is represented as three matrices corresponding to the three red, green, blue (RGB) channels. This is first treated by convolutional filters (D of them in the

figure) that act as follows:

$$(X \star K)_{i,j,k} = \sum_m \sum_n X_{i+m,j+n,k} K_{m,n,k} \tag{2.10}$$

where the input X is a tensor of rank 3 (the third dimension corresponds to the RGB channel) and K the convolutional **filter** which takes the form of a 3-rank tensor of arbitrary dimension. The convolution operation produces a 3-dimensional tensor of dimensions that depend on filter shape and additional options (for instance periodic boundary conditions on the image). In practice, many filters are applied simultaneously to the input image, thereby producing a stack of filtered images as depicted in Fig. 2.11.

The presence of these convolution filters has many consequences: (i) their action is very localized, which is relevant for input data (images) that have short-distance correlations, (ii) convolution is *equivariant*[2] with image translations, which is relevant for object detection since the CNN should not be sensitive to the location of the object within the image, (iii) a more practical advantage is the reduced number of parameters (which scales as the size of the filters and the number of them) compared to a fully-connected neural network of the same input and output dimensions (whose number of parameters scales as the size of the input multiplied by the size of the output). The latter property – sparse connectivity – has enabled CNNs to significantly increase network width and depth without requiring a corresponding increase in training data since the number of free parameters could stay small.

Pooling and final layers. The pooling filters act by reducing the resolution of the image with a coarse-graining operation. For condensed matter physicists, in essence it is very close to a block-decimation operation as seen in renormalization techniques. The aim of pooling operations is to downscale the images to detect features at larger scales, indeed the detection of a specific object should not depend on its size relative to the image size. Thus pooling allows to implement this scale invariance property present in real-world image datasets. Finally, the CNN finishes with a fully-connected part that ensures that the output is of fixed size (since convolution and pooling operations produce output tensors whose dimensions are a function of the input dimension).

The CNNs brought about astonishing improvements in tasks like object recognition, as it became state-of-the-art from 2012 on after winning the ImageNet contest [Krizhevsky *et al.* 2017b]. After that, it was quickly adopted beyond its original use, proving to be very efficient at many other tasks like natural language processing [Yin *et al.* 2017] or in condensed matter physics applications [Carrasquilla & Melko 2017, Choo *et al.* 2019] as we shall see in the next chapters.

2.3.3 Restricted Boltzmann machines (RBM)

Contrary to FFNNs or CNNs, Boltzmann machines are most often used in the context of generative modelling when the task is to approximate a multivariate probability distribution. We will focus here on *restricted* Boltzmann machines which will be the topic of chapter 4. A RBM is an undirected graph containing a layer of **visible** units $\mathbf{v}$ and a layer of **hidden** units $\mathbf{h}$ that are connected as shown in Fig. 2.12.

[2] The convolution operation commutes with any pixel translation.

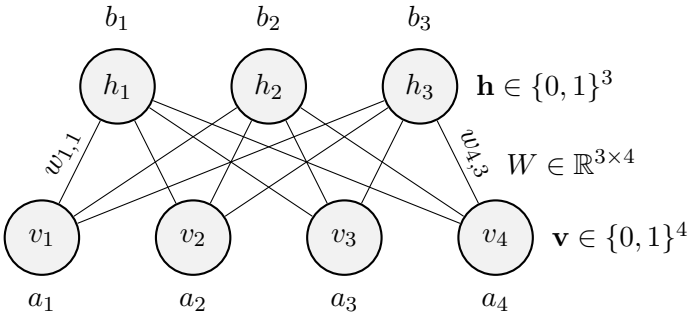


Figure 2.12: Architecture of a restricted Boltzmann machine with a visible layer of 3 units and a hidden layer with 4 units. Note that there are no connections between hidden units nor visible units, hence the name restricted.

It is a so-called energy-based model which encodes the following joint probability distribution:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \tag{2.11}$$

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{v}^T\mathbf{W}\mathbf{h} \tag{2.12}$$

$$Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \tag{2.13}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{W}$ are the free parameters of the RBM.

The normalizing constant Z can in fact be interpreted as the partition function of an Ising-type model with two species of spins interacting with each other via $\mathbf{W}$ and coupled to magnetic fields $\mathbf{a}$ and $\mathbf{b}$. As is commonly the case in statistical physics, Z is not tractable, hence $P(\mathbf{v})$ is intractable when $\mathbf{v}$ and $\mathbf{h}$ are binary vectors. However, one advantage of RBMs is that the conditional probabilities $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$ can be computed exactly. For the interested reader, we refer to [Goodfellow *et al.* 2016] for more details about the training and sampling of RBMs.

2.3.4 Training of neural networks

Many of the progress in the training of neural networks consisted in circumventing limitations arising from the use of gradient descent techniques, the paragraphs below discuss these different issues. Eq. 2.14 shows a schematic cost function $\mathcal{L}$ on a dataset $\mathcal{D}$ and a neural network $f(\cdot;\boldsymbol{\theta})$:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},y)\in\mathcal{D}} \text{dist}\left(y, f(\mathbf{x};\boldsymbol{\theta})\right) \tag{2.14}$$

Stochastic gradient descent. As mentioned in Sec. 2.1.4, minimizing $\mathcal{L}(\boldsymbol{\theta})$ can be done with gradient descent. In ML, a stochastic version of this algorithm called *stochastic gradient descent* (SGD) was designed to speed up learning. Indeed, the cost function in Eq. 2.14 contains a sum that runs over the entire dataset, as a consequence it becomes prohibitive to evaluate it (or its derivatives) at every training step whenever $\mathcal{D}$ is too big. To solve this, the idea is to sample the cost function, i.e, at every training step a random **minibatch** of training samples is picked from the dataset and the total loss function is stochastically estimated on these subsets of data points.

Also mentioned in Sec. 2.1.4 is the paramount importance of the learning rate. Algorithms with adaptive learning rates have been developed and proved to provide significant speedups in deep learning [Duchi *et al.* 2011, Kingma & Ba 2015, Zeiler 2012].

Second-order methods, for instance involving the computation of the Hessian matrix, are typically to be avoided in ML since functions have a very large number of parameters to optimize (and the size of these matrices are as big). In particular, this hinders the use of natural gradient [Amari 1998], which exploits the geometry of the loss landscape. We will discuss in more details the latter algorithm in chapter 4 since it was also proved to be efficient and applicable in the case of variational Monte Carlo studies [Sorella 1998].

Back-propagation. Gradient descent involves the computation of the derivative of the cost function with respect to all the parameters of the neural networks. To do so, the simplest Euler finite-difference scheme would result in evaluating $N_{\boldsymbol{\theta}} + 1$ times different neural networks (where $N_{\boldsymbol{\theta}}$ is the total number of parameters in the NN) at each training step of the gradient descent. The discovery of the back-propagation algorithm [Rumelhart *et al.* 1986] was a major breakthrough in the field as it allows exact and fast gradient calculations by exploiting the architecture of neural networks.

This algorithm consists in a set of recurrent equations that link the vector of activation values $\mathbf{z}^l$ of each layer l and the so-called errors $\boldsymbol{\delta}^l$ to the derivatives of the cost function with respect to any of the parameters of the neural network. We denote $\mathbf{a}^l$ the activation vector of layer l, $\mathbf{z}^l$ being the vector of activation before application of the nonlinear activation function σ, written as follows:

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l \tag{2.15}$$

$$\mathbf{a}^l = \sigma(\mathbf{z}^l) \tag{2.16}$$

with $\mathbf{a}^0 = \mathbf{x}$ is the input vector. The values of $\mathbf{z}^l$ can be obtained from the equations above, which constitutes a *forward pass* in the neural network. For the interested reader, a detailed derivation of the back-propagation equations is provided in [Nielsen 2015]. The vector of errors $\boldsymbol{\delta}^l$ are defined as follows:

$$\boldsymbol{\delta}^l = ((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{l+1}) \odot \sigma'(\mathbf{z}^l) \tag{2.17}$$

with $\boldsymbol{\delta}^L = \nabla_a \mathcal{L} \odot \sigma'(\mathbf{z}^L)$ (where $\mathcal{L}$ is the cost function). Contrary to $\mathbf{z}^l$, the errors $\boldsymbol{\delta}^l$ can be computed only backwards, going from the output layer to the input layer, which explains the term "back-propagation". After only two passing – forward and backward – through the neural network, one can obtain the gradient of the cost using the following relations:

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.18}$$

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{2.19}$$

Preconditioning. As gradient descent is an iterative local algorithm, it is very sensitive to the value of the initial parameters of the optimization. Starting from different initial values of the weights and biases of a neural network, one may get stuck in different local minima (thereby generating variance in the model predictions). Most modern initialization strategies are based on

heuristics and consist for example in sampling a Gaussian distribution with a standard deviation that depends on the neural network architecture [Goodfellow *et al.* 2016].

Learning can become difficult when the loss landscape has a mixture of steep and flat directions since gradient descent treat all directions in parameter space uniformly. One simple trick to avoid this issue is to standardize the data by subtracting the mean and normalizing the variance of input variables (see also Sec. 2.4.1 to decorrelate the input data). By doing this, we ensure that the landscape looks homogeneous in all directions in parameter space [Mehta *et al.* 2019]. This idea was recently generalized in [Ioffe & Szegedy 2015] where normalization is performed on the activations of each layer, which allows for great acceleration of training in deep architectures.

Nature of local minima. Gradient-descent will always find local minima of the cost function, moreover since neural networks are non-convex functions, it is possible that gradient descent eventually gets stuck in local minima with poor generalization performance. Although long considered as a serious hurdle for the success of deep neural networks, practice and theory eventually led to the observation that shallow architectures are more prompt to be trapped in bad local minima compared to deep architectures [Choromanska *et al.* 2015]. This can be explained by the fact that even though the loss surface is extremely rugged and presents many local minima, many of them have good generalization properties. Interestingly, this type of behaviour raised interest from the physics community since it bears similarities with spin-glass models [Geiger *et al.* 2020, Spigler *et al.* 2019].

2.3.5 Regularization

In this section, we focus on explicit regularization strategies for deep neural networks. Many training settings mentioned previously can influence the type of solution we end up with, for example it was shown that for linear models, SGD always converges to a solution with a small norm [Zhang *et al.* 2017], which is an example of implicit regularization.

Parameter norm penalties. As introduced in Sec. 2.2.2.3, applying a penalty in the cost function on the norm of the model parameters is known as weight decay. It is possible to adapt it to neural networks by adding the L^2 norm of the weight matrices (defined as $||W||_2 = \sqrt{\sum_i |w_i|^2}$) to the cost function. Usually the penalty does not apply to the biases because they typically require less data than the weights to converge [Goodfellow *et al.* 2016]. Alternatively, one can choose the L^1 norm ($||W||_1 = \sum_i |w_i|$) which in practice has the effect of setting a subset of the weights to zero for high enough penalty parameter [Goodfellow *et al.* 2016]. This results in a solution that is more **sparse**, which can be used as a **feature selection** mechanism.

Early stopping. When training large models with sufficient representational capacity to overfit, it is often observed that training error decreases steadily over time, but the test error begins to rise again. This means we can obtain a model with better generalization performance by stopping at the point in time where the minimum test error has been reached. This strategy is known as **early stopping**. Its effect is in fact very similar to L^2 regularization [Goodfellow *et al.* 2016].

Dropout. Dropout [Srivastava *et al.* 2014] is a very popular technique consisting in randomly "switching off" neurons (see Fig. 2.13) at each training step with some probability p, the gra-

dient descent is then performed only on the thinned network. This has the effect of preventing co-adaptations of neurons which results in less overfitting. This is very close in spirit to ensemble methods where several models are aggregated to eventually reduce the variance of the predictions, here the different models are obtained with this switching off/on procedure.

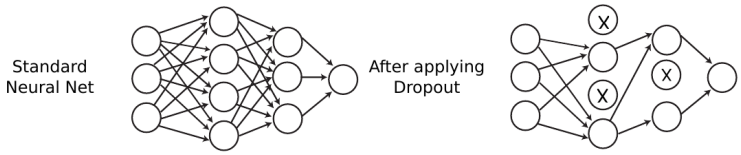


Figure 2.13: Dropout consists in switching off neurons during the training procedure. Figure from [Mehta *et al.* 2019].

Dataset augmentation. As we saw in Sec. 2.2.2.2, more training data allows to improve generalization performance. One popular idea widely used in the field of image recognition is to generate new fake data by slightly modifying the available data. Indeed, transformations like rotation, translation, dilatation, addition of noise of small magnitude are applicable since variations of viewing angle, variations of colors due to shadows or exposition are generally irrelevant for the final classification task.

2.3.6 Software

Many machine learning libraries, in particular Python packages, have been intensively developed over the recent years. The principal ones include Tensorflow [Abadi *et al.* 2015] along with its API Keras [Chollet *et al.* 2015] and the concurrent Torch [Paszke *et al.* 2019]. They provide an optimized set of tools ranging from creation of neural networks, automatic differentiation to GPU parallelism. The existence of these libraries (and others) have clearly helped broadened the impact, accessibility and thus success of ML techniques.

2.4 Dimensional reduction

2.4.1 Principal component analysis

Principal component analysis (PCA) is a linear method used for dimensional reduction, as well as data visualization or analysis. The aim of PCA is to find a new coordinate system where each axis corresponds to a direction of high-variance of the data. The idea is that in many cases, the relevant information in a signal is contained in the directions with largest variance, as depicted for example in Fig. 2.14.

The directions of highest variance can be obtained as the eigenvectors of the covariance matrix of the data. To do so, we first perform singular value decomposition (SVD) on the data matrix $\mathbf{X}$ (where vector of data points $\mathbf{x}$ are stacked to form the matrix), yielding $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{S}$ is a diagonal matrix of singular values s_i, the orthogonal matrix $\mathbf{U}$ (resp. $\mathbf{V}$) contains as its columns the left (resp. right) singular vectors of $\mathbf{X}$, this eventually allows to diagonalize the covariance matrix:

$$\mathbf{\Sigma}(\mathbf{X}) := \frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \tag{2.20}$$

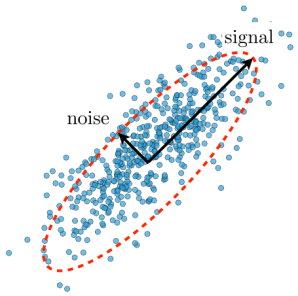


Figure 2.14: Example of a cloud of points where PCA enables to find the largest variance direction (signal) as compared to orthogonal lower variance direction (noise). Figure from [Mehta *et al.* 2019].

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues λ_i sorted in decreasing order. Therefore, to reduce the dimensionality of the input data, we can project the data on the singular components with the largest singular values corresponding to the highest data variance directions. The same idea is central to tensor network techniques used to compress the number of components in quantum wavefunctions in studies of low-dimensional many-body lattice systems.

2.4.2 Variational autoencoders

We briefly introduce in this section a nonlinear dimensional reduction technique based on autoencoders [Kramer 1991]. An *autoencoder* is a neural network that learns to copy its input to its output. As shown in Fig. 2.15, its structure is made of (i) an encoding section (left part of the figure) where the input data is mapped to a *code*, which can be a hidden layer containing generally fewer neurons than the input dimension, (ii) followed by a decoding section where the code is decoded so that it reconstructs the original input. As visible, the code part acts as a bottleneck in the network and the representation of the data is forced to be expressed in a latent space of lower dimension. This is why autoencoders can serve for dimensionality reduction tasks, they have also been used for generative modelling.

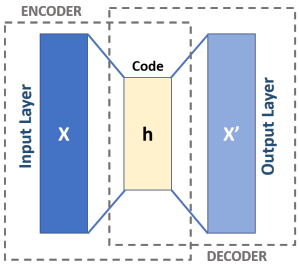


Figure 2.15: Basic architecture of an autoencoder. Figure from [Wikipedia contributors 2020].

2.5 Reinforcement learning

We introduce in this section a machine learning paradigm that has been harvesting a lot of success lately, which is called **reinforcement learning** (RL). We provide in the following a quite general introduction to the main concepts of this field. More details will be given in chapter 5 on the specific kind of RL used in my work.

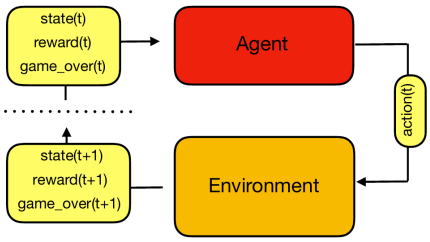


Figure 2.16: In reinforcement learning, an agent and an environment interact with each other. Figure from [Sweke *et al.* 2018].

RL is probably the most natural way to simulate what learning is. As shown in Fig. 2.16, an artificial **agent** (an algorithm) and an **environment** interact with each other: the agent can perform actions that can modify the current state of the environment, in response the latter sends back positive or negative **rewards**, the goal of the agent is then to maximize this reward. The whole difficulty of this game is that some actions will produce an immediate positive high reward from the environment but at the same time they will only result in low reward for the next interactions (for example, the capture of a piece in chess can be poisoned, leading later to higher material loss or checkmate). Consequently, the **policy** of the agent should take into account the long-term consequences of each action. In particular, in any reinforcement learning application, a trade-off between exploration and exploitation needs to be made. On the one hand, the agent has to *exploit* the strategies that proved successful in the past, on the other hand it has also to *explore* uncharted territories to discover new policies that could be more efficient. The dilemma lies in the fact that pursuing only one of the two strategies will inevitably lead to failing at the task.

Reinforcement learning cannot be confused with supervised or unsupervised learning. In supervised learning, an algorithm is provided with examples of solutions of the task, the correct actions are known and easy to obtain from a human expert which is often not the case in RL. Although RL may seem unsupervised, it is also different from unsupervised learning because the reward system implicitly works as a cost function, guiding the learning process towards better performing agents.

2.5.1 A practical example: the game of chess

Reinforcement learning is particularly suited to tasks like game-playing. An important benchmark in this field consists in playing a set of 57 games originally developed for Atari 2600 [Bellemare *et al.* 2015]. A long-standing problem in the RL community was to devise an algorithm that would be able to beat chess and Go human players. This was finally achieved recently in a succession of works [Silver *et al.* 2016, Silver *et al.* 2017, Silver *et al.* 2018] in which

the breakthrough came from a combination of reinforcement learning and deep neural networks.

Figure 2.17: In the game of chess, two players play against each other on an 8×8 square grid. They play one after the other by moving one of his/her 16 pieces to an empty square (black or white) or capture an enemy piece. The winner is the player that was able to capture the enemy king.

The following paragraphs give details on how RL can be used for the game of chess, namely what is the environment, the agent, actions, etc..

- The agent is one of the two players, i.e. it can act on either the white or the black pieces. The possible actions include moving its own pieces to empty squares of the board, capturing an opponent's piece, and so on, while obeying the rules of chess. The action space here is finite and is roughly of the order of 150 possible actions.

- The state of the environment is defined as the positions of all the pieces on the board. The state space is finite but extremely large since there are roughly 10^{50} board configurations in chess (to be compared to 10^{162} configurations for the game of Go). Here as in many other RL tasks, the curse of dimensionality manifests in a severe fashion. Evaluating which action is best in all these game situations is computationally intractable, as a result RL must work by generalization: the agent should be able to deduce an action from its accumulated experience in a new unseen context. The key insight of [Silver *et al.* 2018] to achieve this was to exploit the excellent generalization power of deep neural networks.

- The reward system is a algorithm-dependent, that is it is not defined by the rules of the game. Namely, for chess, a reward is sent at the end of the game, for instance giving 1 point for a win, -1 point for a loss and 0 point for a draw. In that case, a reward is given only after the end of the game which happens after many moves have been made by the agent. This further shows the importance of long-term predictions of the agent, it must be able to anticipate its opponent's moves at the scale of the whole game which can last more than 50 moves.

2.5.2 Mathematical formalization

As we typically deal with discrete time problems, we define variables with a time step index t (as in Fig. 2.16). The action a_t made by the agent on the state s_t of the environment results in a new environment state s_{t+1} and a reward r_t is returned to the agent. These interaction cycles can be formalized as a classical finite Markov decision process governed by the transition probabilities:

$$p(s', r|s, a) = \Pr(s_t = s', r_t = r|s_{t-1} = s, a_{t-1} = a) \tag{2.21}$$

which is the probability to transition from state s at time t to state s' at time $t+1$ under action a, returning the reward r.

The goal of a reinforcement learning agent is to learn a **policy** which maximizes the expected cumulative reward. The decision-policy π of the agent can be modeled by a probability distribution that maps states to probabilities of specific actions, i.e. $\pi(a, s) = \Pr(a_t = a|s_t = s)$ which is the probability that action a is chosen given the system is in state s. The cumulative reward is simply the sum of all rewards obtained while interacting with the environment and it provides a measure of the performance for long-term decisions. This can be put in more quantitative terms with a **value function** $v_\pi(s)$, which defines the value of a state s as the total amount of reward an agent following a policy π can expect to accumulate over the future, starting from that state:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \tag{2.22}$$

with the discount factor $0 \leq \gamma \leq 1$ which allows to vary how much long-term rewards are taken into account. Once this function is known, the optimal policy can be simply obtained as the one that maximizes the value function for all states. However, given the state space is huge in many RL tasks, one needs to resort to an iterative optimization procedure.

Let us sketch how optimization of a policy can be performed with a policy gradient method. The first step is to parametrize the agent's policy as $\pi(a, s) = f_{\boldsymbol{\theta}}(a, s)$, where $f_{\boldsymbol{\theta}}$ can be chosen to be a neural network (which was done in chapter 5). Then, the next step is to design an objective function to optimize with respect to the policy parameters $\boldsymbol{\theta}$. One option is to maximize the value function v_π using a gradient descent algorithm which necessitates to compute the gradient of v_π with respect to $\boldsymbol{\theta}$. Different algorithms exist to compute this gradient using Monte Carlo estimates [Williams 1992] or indirectly in so-called Q-learning [Sutton & Barto 2018]. In chapter 5, we will use a different method that does not rely on gradient computations. The interested reader will find more details about these methods in [Sutton & Barto 2018].

Chapter 3

Automatic classification of phases of matter

Contents

A central focus of condensed matter theory is the understanding of phases of matter. Roughly speaking, a phase is defined by consistent properties a system displays in varying external conditions. It may happen that changing an external parameter abruptly perturbs the system such that its properties radically change: a *phase transition* took place. This chapter deals with the detection of such phase transitions using methods of machine learning (ML).

In Sec. 3.1, we will introduce the general theory of quantum phase transitions and discuss the traditional tools to map out phase diagrams. In Sec. 3.2, we will show how machine learning can be employed to detect *automatically* phase transitions. Sec. 3.3 is devoted to the physics of many-body localization (MBL). We applied these ML approaches to a one-dimensional model in Sec. 3.4 and a two-dimensional model in Sec. 3.5, both exhibiting a – still debated– phase transition. This chapter contains the results we obtained in [Théveniaut & Alet 2019, Théveniaut *et al.* 2020] as well as new results that place emphasis on the interpretation of the neural networks.

3.1 Quantum phase transitions

In general, the study of phases and phase transitions in condensed matter systems can be achieved using mathematical and numerical tools that were specifically designed to tackle these problems. On the analytical side, field theories, mean-field approximation, perturbative expansions or renormalization-group approaches are among the most powerful techniques used by theoreticians to elucidate the existence and stability of phases [Sachdev 2011, Herbut 2007]. On the numerical side, Monte Carlo or variational approaches can take over in models or parameter regimes when analytical calculations become impractical [Becca & Sorella 2017].

3.1.1 Critical points and universality

In strongly correlated systems, we usually consider a lattice Hamiltonian H that encodes the kinetics and interactions of the particles of the system. Hamiltonian H usually depends on external parameters λ like hopping amplitude t, repulsion or attraction strength U or external magnetic fields. Then, the focus is often put on the correlation properties of the ground state but higher-energy states can also be targeted (see Sec. 3.3). Roughly speaking a phase transition is an abrupt change of the characteristics of the state at study upon changing parameter λ. However, $H(\lambda)$ is always a linear function of λ and in most cases[1] it is impossible to observe a non-analyticity of the ground-state energy for finite-size systems. This brings about the first important point that phase transitions are expected to occur only in the *thermodynamic limit*, that is in the limit of an infinite lattice.

It is also important to make other distinctions. First, phase transitions observed as the temperature T is changed are called *thermal* phase transitions. This is to be contrasted with *quantum* phase transitions that occur at $T = 0$ as function of a non-thermal parameter λ, the latter transitions being driven by quantum fluctuations (a manifestation of the Heisenberg uncertainty principle). One may argue that reaching $T = 0$ is not possible in real-life settings, thus this asks the question of the relevance of theoretically studying such a regime. Remarkably

[1] For $H = H_0 + \lambda H_1$ where H_0 and H_1 commute, there can be a level-crossing for the ground-state energy.

enough, a lot of the finite-temperature features can in fact be inferred from the nature of the quantum critical point [Sachdev 2011].

Another important property of phase transitions regards whether they are of first-order or second-order type. During a *first-order* phase transition, the system can absorb or release a fixed amount of energy per volume, they correspond to scenarios in which some observables are discontinuous across the transition. For a *second-order* phase transition, also called continuous phase transitions, the characteristic energy scale vanishes at the critical point. This implies that there exists at least one *diverging length* scale ξ, for instance a correlation length or a localization length. It is possible to characterize more finely the properties of the system in the vicinity of critical points with *critical exponents* by defining a correlation length exponent ν as follows (with λ_c the critical value):

$$\xi \sim |\lambda - \lambda_c|^{-\nu} \tag{3.1}$$

The characterization of phases of matter is often done based on the correlations of the state. A powerful method pioneered by Landau was to identify a quantity called *order parameter*, that would take a finite non-zero value in the ordered regimes (long-range correlations) and be zero in the disordered phase (short-range correlations). For example, this applied well to the classical 2D Ising model where the total magnetization is the order parameter of the ferromagnetic-paramagnetic transition as it takes a non-zero finite value in the ferromagnetic phase (since most spins are pointing up/down) but vanishes in the paramagnetic phase (since spins' orientation is random). The Landau theory of phase transitions also put forward the mechanism of symmetry-breaking where some symmetries of the system are not preserved across a phase boundary. Although powerful, this hypothesis does not hold for the –later discovered– topological phase transitions [Schirber 2016].

An important discovery in the field was that of *universality*. This states that the critical behaviour of a physical system only depends on the dimensionality and the symmetries of the model, which means that there are physical systems which, regardless of their microscopic properties (lattice geometry, chemical constituents, etc..) do behave similarly near a phase transition. This involves that they have the same critical exponent ν and/or other critical exponents associated to observables like specific heat, magnetic susceptibility. For example, the ferromagnetic-paramagnetic transition of magnets is in the same universality class as the liquid-gas critical point in water. Universality in fact results from the divergence of the characteristic length scale at the critical point which makes finite length scales irrelevant. This also has the consequence of making the system scale-invariant at the critical point, which is incidentally the basis of the renormalization group theory [Wilson 1975].

3.1.2 The finite-size scaling method

The divergence of the critical length ξ will never be observed in practice in simulations of continuous transitions on finite-size systems. Instead, ξ will be cut off at the size L of the system, which means that near the critical point we have $L \sim \xi \sim |\lambda_c(L) - \lambda_c|^{-\nu}$ where $\lambda_c(L)$ is the finite-size pseudo critical point and λ_c is the true critical point in the thermodynamic limit. Similarly, an observable that is expected to diverge at the transition will remain finite for finite-size systems. From these observations, it is possible to derive the following expression for

an extensive observable $\mathcal{O}$ associated to critical exponent β [Newman & Barkema 1999]:

$$\mathcal{O}(\lambda - \lambda_c) = L^{\beta/\nu}\tilde{\mathcal{O}}\left[L^{1/\nu}(\lambda - \lambda_c)\right] \tag{3.2}$$

where $\tilde{\mathcal{O}}$ is a dimensionless function that does not depend on L. For an intensive quantity, the factor $L^{\beta/\nu}$ would be absent. By explicitly writing the finite-size behaviour (dependence with L) of observables, one can determine the critical exponents ν and β by a collapse procedure.

3.2 Automatic classification of phases of matter

The traditional way of studying phases and phase transitions relies on the choice of observables that can be motivated by their experimental accessibility (like specific heat) or theoretical considerations like the identification of order parameters. Recently, a parallel was drawn between the study of phases of matter and the field of data analysis. The state of a physical system is described by the state of its many constituents which makes up a data of large dimension, one can then see that measuring the value of an observable on this state corresponds to a projection of this large dimension data (the state) onto a low-dimensional space (here of dimension 1, the measurement). Following up on this observation, several approaches based on machine learning were developed to tackle the problem of mapping phase diagrams.

After highlighting the specificities of data in the physics context in Sec. 3.2.1, we introduce three methods using supervised, unsupervised and semi-supervised learning. We will discuss in details the pros and cons of these ML-based approaches in Sec. 3.2.5.

3.2.1 Data in condensed matter physics.

It is important to clarify which data will be considered in the following sections. First, data can be *experimental*, i.e. directly obtained from experimental measurements or *synthetic*, i.e. generated from numerical simulations of a model. We will focus on the latter type in the rest of the chapter. Second, it is not obvious which "image" of a physical system is relevant for a machine learning treatment. For classical spin models like the Ising model, it is natural to encode real-space spin configurations as colored pixels with the mapping: $\uparrow = \blacksquare$, $\downarrow = \square$. For a two-dimensional square lattice, the representation is even more straightforward as can be seen in Fig. 3.1.

For a quantum system, the first difficulty arises from the exponential complexity of the quantum wave-function. Indeed the description of a quantum state needs to specify a number of coefficients that grows exponentially with the system size. In addition to that, the notorious curse of dimensionality in ML entails that the size of the dataset (number of samples) has to increase exponentially with the dimension of the data samples to ensure efficient training. These two phenomena may severely limit the treatment of large system sizes. Additionally, a quantum state can be decomposed in many different basis sets, which adds some freedom to their representation. To limit the exponential complexity of the wave function, one can compress the quantum state in a more compact form [Broecker *et al.* 2017b, van Nieuwenburg *et al.* 2017, Beach *et al.* 2018, van Nieuwenburg *et al.* 2018, Zhang & Kim 2017, Théveniaut & Alet 2019, Théveniaut *et al.* 2020] like preprocessing techniques in ML. Preprocessing data can be crucial to achieve training, however in the physics context this step can induce biases as we will discuss later in Sec. 3.4.1.

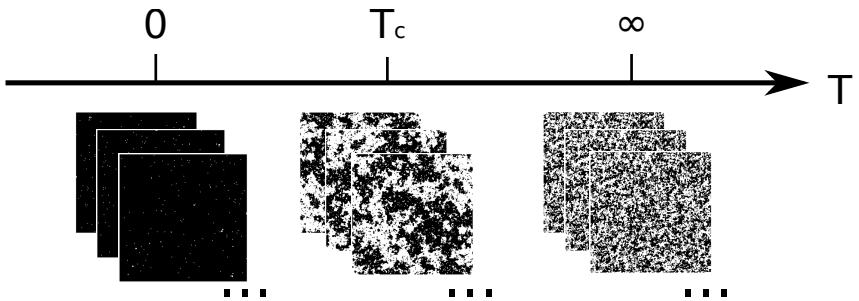


Figure 3.1: Spin configurations of the two-dimensional Ising model on the square lattice representative of different temperature regimes (ferromagnetic, critical, paramagnetic from left to right) viewed as black-and-white images.

Obtaining large datasets is straightforward when data is generated from Monte Carlo simulations since obtaining more data points simply mean sampling more. It is important to keep in mind that the quality of the data is key to the success of these ML approaches, therefore autocorrelation between Monte Carlo samples for instance are of significance for the interpretation of the final results. An interesting property of disordered systems is that the dataset can be enlarged very simply by generating more instances of the system from different random realizations of disorder. This remark applies as well for ground states of classical models that have a very large degeneracy. On the contrary, for models without disorder, at zero temperature, one can be faced with the problem of having too small datasets since there are not enough available physical "images" representative of a given phase.

3.2.2 Supervised method

Carrasquilla and Melko proposed a method based on supervised learning [Carrasquilla & Melko 2017] that works as follows: (i) physics knowledge is used to label training samples in well understood limits of the phase diagram i.e. each training sample has a *phase label* (see red training regions in Fig. 3.2), (ii) then a neural network is trained to classify accurately these training samples according to their phase label, (iii) finally the trained NN is used to predict the phase label of samples across the whole phase diagram and its predictions are used to map out the phase boundaries.

Carrasquilla and Melko show that, trained on classical spin configurations labelled as ferromagnetic (low-temperature) and paramagnetic (high-temperature) generated from a Monte Carlo simulation of the square lattice Ising model, a neural network is capable of not only performing well on the training dataset but also allows to predict a critical temperature as well as a correlation length exponent ν consistent with the exact known values (the latter being obtained with a finite-size scaling analysis on the neural-network output). Quite remarkably, they also show that a NN trained on square lattice configurations can accurately predict the phase boundaries of an Ising model defined on a *triangular* lattice geometry. By designing a toy model reproducing the NN predictions, they found out that the NN in fact computes the total magnetization of the configuration, which is known to be the order parameter of this transition.

Next, they turned to more difficult scenarios where no conventional order parameter exists. They show that a NN is able to distinguish the ground states and the high-temperature

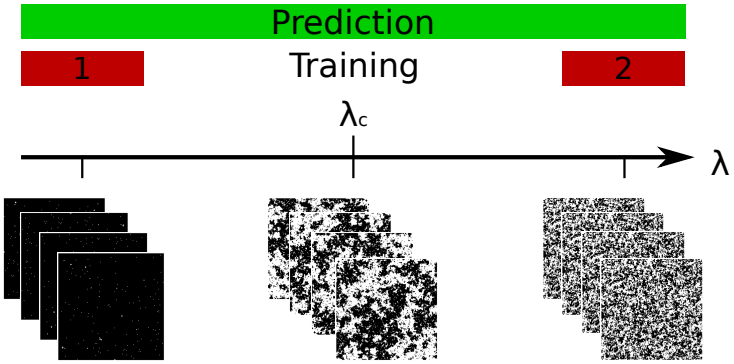


Figure 3.2: The supervised learning approach works by training a neural-network to distinguish samples from phases 1 and 2, physics knowledge is used to label the training samples originating from the limits of the phase diagram. Phase prediction is done on all samples across the phase diagram.

states of the two-dimensional frustrated square-ice Hamiltonian. Contrary to the ferromagnetic-paramagnetic transition in the Ising model where the two phases could be told apart to the naked eye, here it is much harder to tell since the distinction lies in the decay of spin-spin correlations changing from power-law (at $T = 0$) to exponential (at $T = \infty$). They were also able to classify accurately low-temperature from high-temperature states of an Ising lattice gauge theory, which is an example of topological phase transition with no order parameter, though requiring the use of a deep CNN. The discriminative power of the CNN was explained by the existence of convolutional filters each detecting satisfied local energetic constraints.

3.2.3 Unsupervised method

Concurrently, Wang [Wang 2016] and Wetzel [Wetzel 2017] treated the same problem employing unsupervised learning techniques. Compared to the previous method, this approach has the advantage that it does not require the labelling of "images" and hence works without assuming the existence of a phase transition. Using PCA (see Sec. 2.4.1 in chapter 1), they show that a majority of the variance of the spin-configuration data matrix (consisting of stacked spin configurations from all temperatures) was carried by the first principal axis, which was in one-to-one correspondence with the magnetization. Wang studied a spin model conserving the total magnetization and was able to uncover a seemingly unknown order parameter for the transition occurring in this model.

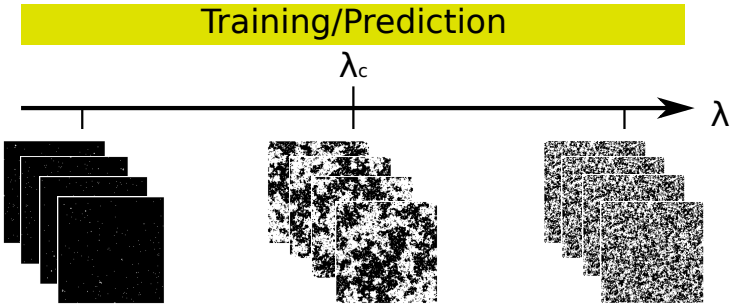


Figure 3.3: The unsupervised learning approach works by applying data analysis tools (PCA, clustering, etc..) on the set of samples coming from the whole phase diagram.

3.2.4 Semi-supervised method

An alternative approach was proposed by van Nieuwenburg, Liu and Hubert in [van Nieuwenburg *et al.* 2017] and at the same time by Trebst and coworkers in [Broecker *et al.* 2017a]. There, contrary to the supervised method, the training dataset contains samples not only from the extremes but across all parameter regimes (see Fig. 3.4). The crucial insight is that it is possible to apply a labelling hypothesis for which labels are assigned to samples with no physical ground. In most cases, data can be ordered by the variable driving the transition (for instance temperature in the ferromagnetic-paramagnetic transition of the Ising model), this means that a labelling hypothesis is achieved by assigning label "1" (resp. "2") to all samples corresponding to $\lambda \leq \lambda_c^{\text{hyp}}$ (resp. $\lambda \geq \lambda_c^{\text{hyp}}$) as shown in Fig. 3.4 where λ_c^{hyp} is the hypothetical "critical" point. For each hypothesis λ_c^{hyp}, a neural network is trained to learn the data labelling. Two situations may arise: on the one hand, for points B and C in Fig. 3.4 the samples do not belong to the same phase but have the same phase label, therefore we expect difficult training. On the other hand, for points A and B in Fig. 3.4 the samples belong to the same phase but are labelled differently, in this case we expect poor prediction performance since the neural network will predict both samples to be of the same type whereas the labelling hypothesis says the opposite. By testing different labelling hypothesis, their performance can be tracked as a function λ_c^{hyp} and is expected to take a generic W-shape as argued in [van Nieuwenburg *et al.* 2017]. The best performing hypothesis is expected to coincide with the actual phase labelling (in other words $\lambda_c^{\text{hyp}} = \lambda_c$), hence unravelling the phases and phase boundaries.

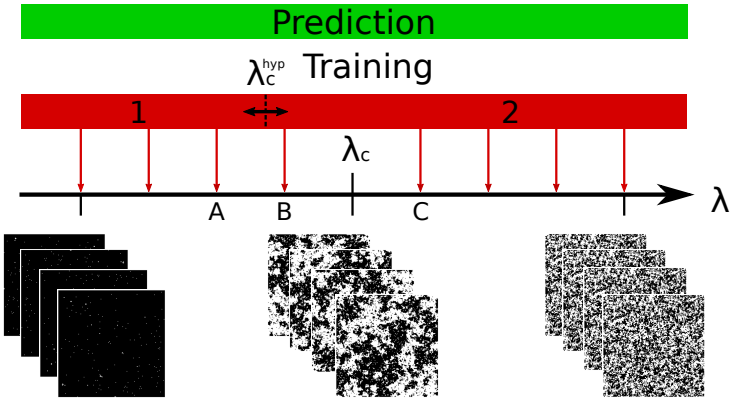


Figure 3.4: The semi-supervised approach works by first assuming a labelling hypothesis λ_c^{hyp}, samples lying for which $\lambda \leq \lambda_c^{\text{hyp}}$ (resp. $\lambda \geq \lambda_c^{\text{hyp}}$) are labelled as phase 1 (resp. phase 2). Then, training is attempted and the training performance is reported as a function λ_c^{hyp}. The transition point is detected with the hypothesis of maximum training accuracy, indeed when $\lambda_c^{\text{hyp}} = \lambda_c$ training is expected to be the easiest.

3.2.5 Advantages and drawbacks

The machine-learning approaches exposed above constitute a complete paradigm shift in the field. Rather than using human knowledge and intuition to identify order parameters or relevant observables, these methods work by presenting raw data from a physical system to a learning algorithm that will automatically learn the most relevant features, which are characteristic of the phases at play. We list below the advantages of these methods:

- **Great versatility.** These methods work without the knowledge of the Hamiltonian which means they can be applied in principle in any context: classical or quantum models, frustrated, bosonic or fermionic systems in any dimension, for topological phase transitions [Beach *et al.* 2018, Suchsland & Wessel 2018] in equilibrium [Ch'ng *et al.* 2017] or out-of-equilibrium [Schindler *et al.* 2017, van Nieuwenburg *et al.* 2018]. This is also of great interest for experimental settings where the exact Hamiltonian is not known.

- **NNs see better than humans.** These methods can discern intricate patterns, find subtle correlations in the data, that are otherwise invisible to human eyes. This is of particular relevance for experimental data that are often plagued with noise and other imperfections. NNs can also perform hypothesis testing as was shown in [Bohrdt *et al.* 2019, Zhang *et al.* 2019b] where NNs can see the compatibility of single experimental snapshots of density-matrix with respect to different theoretical predictions which cannot be distinguished by conventional observables. If the NN has access directly to the state or snapshots (projective measurements), it provides an unbiased predictor and can in principle allow to rule out theories [Greitemann *et al.* 2019] or even used to discover new physical theories [Iten *et al.* 2020].

- **Better phase transition diagnostics.** One possibility is that NN learns an observable that is less sensitive to finite-size effects than other human-engineered quantities, which can be crucial for for certain models with random disorder [Khemani *et al.* 2017,

Panda *et al.* 2020]. Some works also claim that NNs can provide sharper estimates of critical points [Venderley *et al.* 2018, Rem *et al.* 2019] and that has less variance with the same number of samples compared to some conventional observables [Huembeli *et al.* 2018].

- **A tool for material design.** If measurements are costly at different locations of the phase diagram, one can use this approach to map out phase diagram in a fast way to suggest interesting experimental conditions for measurements. One can use it as a material design technique where physical models are optimized to increase the critical temperature of high-T_c superconductors for instance, in a systematic and fast way using NNs.

- **Good scaling properties of NNs extending the range of accessible regimes.** The compositional nature of NNs allow to treat hierarchical data (for instance patterns of spins at different scales) in an exponentially efficient way. This paves the way for descriptions of physical states that live in exponentially large state space from data (input dimension) and neural network (number of parameters) scaling only polynomially with system size. It is notorious that certain observables that might be crucial for the understanding of certain phase transitions are extremely hard to compute such as off-diagonal observables in QMC, or the entanglement spectrum that is both experimentally and numerically exponentially difficult to obtain. Therefore NNs provide a new route towards finding scalable and relevant observables for general phase transitions.

- **Possibility of knowledge transfer.** The generalization power of neural networks offer an exciting path towards the mapping of phase diagrams in regimes that are hard to simulate numerically. The ML approach allows for extrapolating the predictions of neural networks to physical models of different geometries [Carrasquilla & Melko 2017], larger sizes [Théveniaut & Alet 2019, Saraceni *et al.* 2020] or to phases that cannot be accessed by current techniques like regimes where there is a sign-problem [Broecker *et al.* 2017b, Vargas-Hernández *et al.* 2018].

It is important to keep in mind that these approaches *a priori* also suffer from the pitfalls of any NN-based methods regarding data (too small amount, biased), neural networks (opacity of the decision function, vulnerability to pixel attacks) or training (overfitting, underfitting,..).

3.3 Many-body localization

Anderson localization is a quantum phenomenon occurring in random media where one electron can become localized in real space due to destructive interferences of its own quantum paths. Investigating the fate of localization when electrons are interacting with each other led to the discovery of many-body localization, dating back to the seminal contributions of Gornyi, Mirlin and Polyakov [Gornyi *et al.* 2005] in 2005, Basko, Aleiner and Altshuler [Basko *et al.* 2006] in 2006. In these works, these authors were able to show with perturbative arguments that localization can resist weak interaction at high temperature. Later Huse and coworkers [Oganesyan & Huse 2007, Pal & Huse 2010] discovered a metallic-to-insulator phase transition at infinite temperature as a function of disorder strength in the presence of interactions.

3.3.1 Thermalization in closed systems

Many-body localization is in fact related to more fundamental questions such as the nature of the dynamics of an *isolated* interacting quantum system. Given a quantum state $|\Psi\rangle$, its evolution is fully governed by the Hamiltonian H according to the following equation (solution of the Schrodinger equation):

$$|\Psi(t)\rangle = e^{-iHt}|\Psi\rangle \tag{3.3}$$

By diagonalizing the Hamiltonian, one can rewrite $|\Psi\rangle$ in terms of the eigenvectors $|E_n\rangle$ as follows $|\Psi\rangle = \sum_n c_n|E_n\rangle$. As a result, Eq.3.3 transforms into:

$$|\Psi(t)\rangle = \sum_n c_n e^{-iE_n t}|E_n\rangle \tag{3.4}$$

The expectation value of an observable $\mathcal{O}$ can then be expressed as:

$$\langle\Psi(t)|\mathcal{O}|\Psi(t)\rangle = \sum_{n,m} c_n c_m^* \mathcal{O}_{mn} e^{-i(E_n - E_m)t} \tag{3.5}$$

One could expect from Eq.3.5 that the diagonal terms $\mathcal{O}_{nn}$ in the eigenbasis dominate at long times since the contribution coming from terms where $E_n \neq E_m$ will dephase out upon summation and eventually vanish at long times. In the thermodynamic limit, one may however argue than in a many-body system the energy gaps close exponentially fast, which could counteract the dephasing. To justify more clearly the predominance of the diagonal ensemble, one often refers to the so-called Eigenstate Thermalization Hypothesis (ETH) that states that:

$$\langle E_n|\mathcal{O}|E_m\rangle = \overline{\mathcal{O}}(E)\delta_{nm} + e^{-S(E)/2} f(E,\omega) R_{nm} \tag{3.6}$$

where $S(E)$ the microcanonical entropy, $\overline{\mathcal{O}}(E)$, $f(E,\omega)$ are smooth functions of their arguments with $E \equiv (E_n + E_m)/2$ and $\omega = E_n - E_m$ and R_{nm} are random independent variables with zero mean and a unit variance. In particular, this relation means that eigenstates that are close in energy have the same properties (e.g. local observables value) and form a microcanonical ensemble. The ETH is found to be satisfied in most many-body quantum models.

A system that is many-body localized (MBL) does not obey ETH anymore, it does not thermally equilibrate under its own dynamics. This has many consequences on the nature of MBL systems: they exhibit Poisson-type energy level statistics, they are lowly entangled even at very high energy (unlike the volume-law expected for high-energy states), integrability emerges in the form of local integrals of motions, the memory of initial conditions persists at arbitrary long times, etc. A full review of the features of the MBL phase and of the critical phenomena in the transition region towards ETH regimes goes beyond the scope of this thesis and we refer to recent reviews for a more detailed discussion [Abanin *et al.* 2019b, Abanin & Papić 2017, Alet & Laflorencie 2018, Nandkishore & Huse 2015].

3.3.2 The random-field Heisenberg model

One of the paradigmatic model hosting a ETH-MBL transition is the Heisenberg model with on-site random magnetic fields. It has the following form:

$$H = \sum_{i=1}^{L} \mathbf{S}_i \cdot \mathbf{S}_{i+1} - \sum_{i=1}^{L} h_i S_i^z \tag{3.7}$$

where $\mathbf{S}_i = \frac{1}{2}\boldsymbol{\sigma}_i$, $\boldsymbol{\sigma}_i$ are Pauli matrices and h_i are random fields uniformly drawn in a box $[-h, h]$ with h the disorder strength. It was studied numerically in [Luitz *et al.* 2015] where numerous properties of the two phases were probed like spectrum statistics through gap ratios, entanglement properties through entanglement entropies, and Hilbert space localization studied with participation entropies whose scaling is shown in Fig. 3.5. Finite-size scaling on each of these observables has been performed resulting in estimates of the critical point around $h_c \approx 3.7$ and localization length exponent $\nu \approx 1$ in the middle of the spectrum ($\varepsilon = 0.5$).

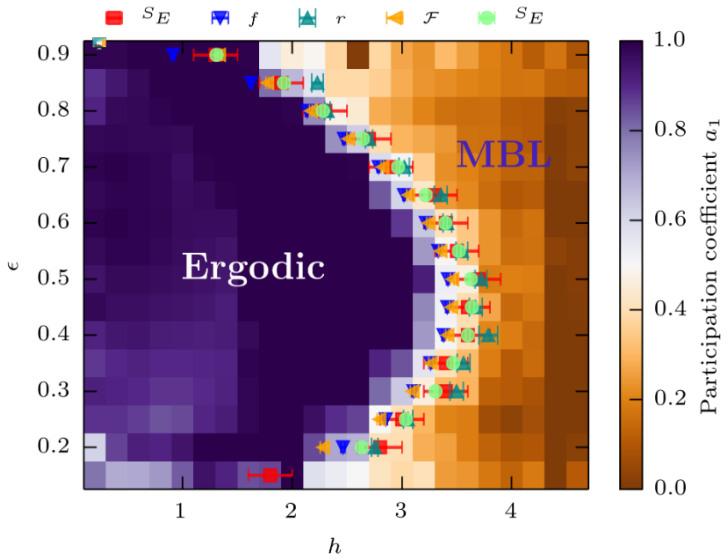


Figure 3.5: Disorder (h) - Energy (ε) phase diagram for the random field Heisenberg model [Luitz *et al.* 2015].

A numerical challenge. Due to the specificities of the MBL problem, its numerical treatment is very challenging. We refer to [Macé & Alet 2019] for a thorough discussion on the difficulties specific to the MBL problem. Let us enumerate a few of these issues: (i) MBL concerns high-energy eigenstates, which are not the targets of most traditional techniques like variational Monte Carlo, (ii) most Quantum Monte Carlo algorithms are inoperative since they assume the existence of a heat bath whereas systems considered here are isolated, (iii) the entanglement scaling is different in ETH (volume-law) and MBL (area-law) regimes which prevents the use of many variational Ansatz which are designed to approximate states that have area-law entanglement, as a consequence matrix-product states would be for instance inoperative in the ETH regime.

Open questions and current challenges. Despite a decade of intense work, many aspects of the MBL problem evade a full theoretical understanding. One of the most prominent question is its very existence. Although it is widely considered to be established in dimension 1, there has been recent polemics [Suntajs *et al.* 2020, Abanin *et al.* 2019a] that highlighted the strong finite-size effects at play in these types of transitions [Panda *et al.* 2020]. The existence of MBL phases in larger dimension is still an open problem. Another very debated question is the universality

class of the ETH-to-MBL transition, since exact numerics are very sensitive to finite-size effects, initial works based on phenomenological renormalization group equations led to different descriptions of the critical region with critical exponent $\nu \approx 3$ [Potter *et al.* 2015, Vosk *et al.* 2015], while recent improvement point towards a BKT-type transition [Dumitrescu *et al.* 2019, Goremykina *et al.* 2019, Morningstar & Huse 2019, Morningstar *et al.* 2020]. Another subject that is not fully addressed is the existence of a many-body mobility edge the thermodynamic limit (that is, the transition point depends on the energy density). Although numerics indeed show such an edge (see Fig. 3.5), there are theoretical arguments precluding their survival in the thermodynamic limite [de Roeck *et al.* 2016].

3.4 Application to the ETH-MBL transition in a 1D model

The current limitations of numerical techniques to tackle the MBL problem and the promising application of machine learning in physics motivated works using the ML machinery on MBL systems. In particular, the debate regarding the existence of an order parameter of the transition offers an interesting benchmark to explore the ML abilities to treat such a difficult transition. Moreover, the strong finite-size effects present in random disorder MBL models [Khemani *et al.* 2017, Panda *et al.* 2020] calls for the use of observables that are less sensitive to finite-size effects, which could be found via a ML procedure (as discussed in Sec. 3.2.5).

As a result, a number of works has been devoted to the study of ETH-MBL transitions in one-dimensional disordered quantum systems and were able to achieve (i) qualitative location of MBL and ETH phases in agreement with more conventional approaches and (ii) detection of "new" phases in known models [Venderley *et al.* 2018, Hsu *et al.* 2018]. Although quite successful, we find that these works also have the following limitations:

- the detection of a phase transition is sometimes claimed considering small systems without attempting a finite-size scaling,
- training is done on human-processed data like the entanglement spectrum of eigenstates, which might bias the learning process,
- the influence of hyperparameters is not systematically examined,
- the interpretation of the neural network is sometimes absent,
- the existence of a first estimate of the transition point ($h_c \approx 3.7$ in model (3.7)) can bias some work towards claiming the success of machine learning methods as soon as this value is recovered.

Our work was an attempt to fill the above-mentioned gaps and we provide an ML-based extensive study of the MBL-to-ETH transition of model (3.7) following the guidelines:

1. *Minimizing human intervention* to reduce the bias that can possibly come from data preprocessing, choice of specific NN architecture, *ad hoc* interpretation of the NN output, etc. We want to remain as agnostic as possible regarding the physics at play, which could possibly reveal new phenomena in the transition. As a byproduct, the more general the input is, the more applicable it will be to other phase transitions.

2. *Scalability*, the input data dimension should be reduced so that training of a neural network remains tractable for the largest system sizes we have ($L = 24$) since we want to perform a finite-size scaling analysis.

3. *Minimizing the influence of unphysical parameters*, the variability of the results with respect to the method parameters (data formatting, NN hyperparameters) should be investigated and mitigated as much as possible.

4. *Interpretability*, data and NN architecture should allow for a possible interpretation of the learned decision function.

These guidelines may sometimes be antagonist, the following sections present the compromise we ended up with to match all these demands. Additionally, we provide a PCA of the dataset in Sec. 3.4.2 and interpretation of the neural networks in Sec. 3.4.6, which did not appear in our published work [Théveniaut & Alet 2019].

3.4.1 Description of the dataset

The following section is a rather lengthy discussion where we evaluate the pros and cons of different types of input data and justify our final choice of input data.

Which data? A variety of data types have been considered in the previous related works. Entanglement spectrum of eigenstates have been used in [van Nieuwenburg *et al.* 2017, Hsu *et al.* 2018, Schindler *et al.* 2017, Venderley *et al.* 2018, Durr & Chakravarty 2019], dynamical observables in [Doggen *et al.* 2018, van Nieuwenburg *et al.* 2018], eigenergies in [Rao 2018, Kausar *et al.* 2020] and full eigenstates in [Zhang *et al.* 2019a, Huembeli *et al.* 2019]. The specificities of the ETH-MBL transition certainly motivated these choices since entanglement scaling with system size, observables decay or growth with time and energy level statistics are known to be different in both phases. We think that the last approach, taking eigenstates as input data, is unbiased in the sense that all information is stored in the eigenvector (all quantities above can be obtained from eigenvectors) and therefore the most promising regarding possible new insights in the physics at play. Additionally, it is known that thermal eigenstates can fully encode a local Hamiltonian [Garrison & Grover 2018, Qi & Ranard 2019], therefore a neural-network provided with such an eigenstate could in principle know the Hamiltonian of equation (3.7). On the MBL side however, the authors of [Dupont *et al.* 2019] showed that there is not a one-to-one correspondence from MBL eigenstates to Hamiltonian.

Choosing full eigenstates as input data raises issues. First, one has to expand these states in a basis set, which leaves room for human –undesired– intervention. Our choice was to stick with the S^z computational basis which is the physically-relevant one in the infinite disorder limit as it diagonalizes the model. This allows us to keep our approach *scalable* and have access to the largest system sizes (due to sparsity of the matrix). An unbiased solution could be to choose a random basis set, however expressing the eigenstates (obtained in the S^z basis) in a random basis involves multiplying each of them with a fully filled matrix of exponentially-increasing dimension, which is clearly prohibitive.

Given an eigenstate $|\Psi\rangle$ and its expansion in the S^z basis $|i\rangle$, $|\Psi\rangle = \sum_i c_i|i\rangle$, we denote its

real[2] amplitudes as c_i and the probabilities as $p_i \equiv |c_i|^2$. If one tries to input the the whole eigenstate as the vector $(p_0, \ldots, p_{2^L-1})$, the input dimension would eventually grow exponentially with system size (for instance an eigenstate of the largest considered system in our work contains 2.8 millions of coefficients, for $L = 24$). This is problematic regarding the tractability of our approach since the curse of dimensionality of ML states that the size of the training dataset should scale exponentially with the input dimension, as a result memory or long training issues would be inevitable.

Which preprocessing? Compressing a quantum many-body state is arguably the biggest challenge of strongly correlated physics. Unfortunately, as discussed in Sec. 3.3.2, the nature of the transition prevents us from applying variational approximation of the eigenstates with ansatz like MPS and therefore there is no obvious compressed parametrization of these states that we could use as input data.

As a consequence, we chose to compress eigenstates by hand: our solution is to keep the N_c largest probabilities p_i for each eigenstate, shown for illustration in Fig. 3.6 for $N_c = 256$ and different L. Note that the basis states associated to the largest coefficients differ from one eigenstate to the other. Instead of using the probabilities p_i, we could directly use the real amplitudes c_i with the additional sign information. We performed this analysis and reported the results in appendix C of [Théveniaut & Alet 2019]. Surprisingly, even though the amplitudes contain in principle more information than the probabilities, we were not able to show any advantage of this choice of input data. This formatting introduces the *truncation order* N_c parameter whose influence will be investigated in Sec. 3.4.6. A crucial advantage of keeping only the largest probabilities is that *it allows to have data from different systems sizes L of same dimension N_c.*

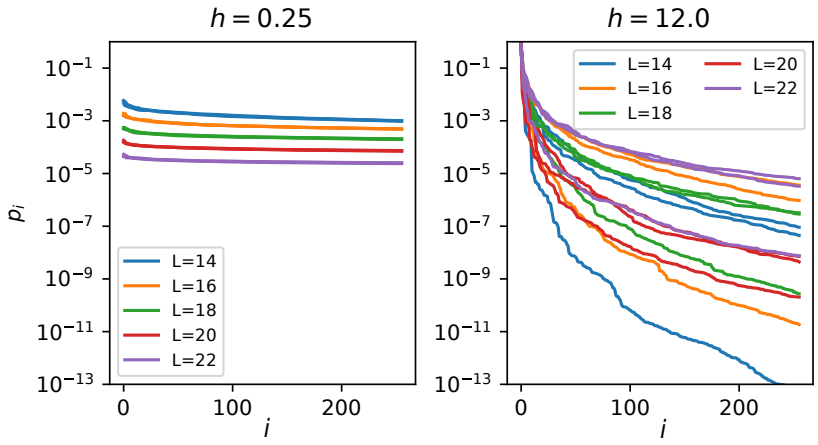


Figure 3.6: Examples of $N_c = 256$ highest probabilities p_i for eigenstates in the middle of the spectrum ($\varepsilon = 0.5$) for different disorder realizations and system sizes for two disorder values located strongly in the ETH (left) and MBL (right) phases. Figure from [Théveniaut & Alet 2019].

[2] because the Hamiltonian (3.7) is real and symmetric (up to degeneracies which can occur only exceptionally due to the random part in the Hamiltonian).

Possible bias? Our understanding of model (3.7) gives us strong confidence that this compression step, although somewhat brutal at first sight, preserves many physically relevant features. Indeed, on the MBL side, in the infinitely strong disorder limit, eigenstates are product states in the S^z basis which translates into having one $p_i = 1$ while all the other basis states have $p_{j\neq i} = 0$. As first shown in [Serbyn *et al.* 2013], the structure of MBL states is robust as disorder is decreased and local conservation laws emerge where the Hamiltonian can be rewritten in terms of pseudo-spin operators that are dressed versions of the original spin operators. The physical extension of these local integral of motions decreases exponentially with distance to the localization center. Eigenstates are then product-states in this pseudo-spin basis and by construction, their amplitudes spread over S^z basis states around the localization center. Therefore we also expect a dominant component p_i at finite but large disorder strength. On the ETH side, one expects a random coefficient structure with no correlation between basis states, thus the probabilities should distribute uniformly.

It was shown [Luitz *et al.* 2015, Kjäll *et al.* 2014, Luca & Scardicchio 2013] that quantities computed from the p_is like the inverse participation ratio $\mathrm{IPR}(|\Psi\rangle) = \sum_i p_i^2$ or participation entropies $S_q(|\Psi\rangle) = \frac{1}{1-q}\log\left(\sum_i p_i^q\right)$ are discriminative of the MBL and ETH phases. These observables are most sensitive to the largest probabilities p_i (independently of which basis state corresponds to index i, one eigenstate from another). By removing the reference to which basis set each p_i comes from, we loose all information about the structure of eigenstates in Hilbert space. It was shown [Luitz *et al.* 2015, Macé *et al.* 2019] that eigenstates are in fact delocalized in Hilbert space in both regimes (though with a much lower slower growth of participation entropies for MBL). One can suspect that giving the probabilities p_i as input data will bias the ML analysis to find quantities that are easily expressed as a function of p_i, like IPR or participation entropies. It seems less likely that the procedure will uncover order parameters that resemble gap ratios [Oganesyan & Huse 2007].

An alternative formatting. An alternative approach which we did not explore is to sample each S^z basis state $|i\rangle$ with probability p_i and store these samples. As a result, sampling N times a given eigenstate would give rise to a data sample of size $N \times L$ (since $|i\rangle$ is defined by L quantum numbers). This formatting has the advantage of keeping the input dimension polynomial w.r.t system size L, as well as preserving the Hilbert space structure of eigenstates. It is well suited for models where QMC works since this is precisely the type of data generated by these techniques. However, the representation of the same state can fluctuate due to random sampling of the basis states, which may pose problems. Lastly, this formatting may be more prompt to capture quantities like local magnetizations which are discriminative of the transition [Pal & Huse 2010, Laflorencie *et al.* 2020] since the input data is itself made of the local spin magnetizations.

Dataset details. We obtain exact eigenstates at energy density $\varepsilon = 0.5$ (infinite temperature) of model (3.7) with the shift-invert method [Pietracaprina *et al.* 2018b]. We use periodic boundary conditions and consider eigenstates in the $S^z = 0$ sector (the total magnetization $S^z = \sum_r S^z_r$ is conserved in this model). We insist on having a large, state-of-the-art dataset. For training, we use 1000 realizations of disorder per disorder strength and 250 (respectively, about 150) realizations of disorder at prediction time for sizes $L = 14, 16, 18, 20, 22$ (respectively, $L = 24$). For each realization of disorder, we compute 100 (respectively, 60) eigenstates for $L \leq 22$ (respectively, for $L = 24$). We use a fine grid of disorder strength, specially close to

the alleged transition region.

3.4.2 Dataset analysis with PCA

Data analysis usually begins by studying the dataset with the help of the linear methods such as PCA (see Sec. 2.4.1 of chapter 2). This analysis was not performed in our work [Théveniaut & Alet 2019] and for simplicity, we will only treat data originating from a $L = 16$ spin chain. The PCA is done on a matrix consisting of 1300 truncated eigenstates (stacked on top of each other) where the first $N_c = 128$ highest probabilities are kept, originating from 13 different disorder strengths and 100 different disorder realizations per disorder. We will also study the effect of randomization ($p_i \leftarrow p_{\mathcal{P}(i)}$ with $\mathcal{P}$ a permutation).

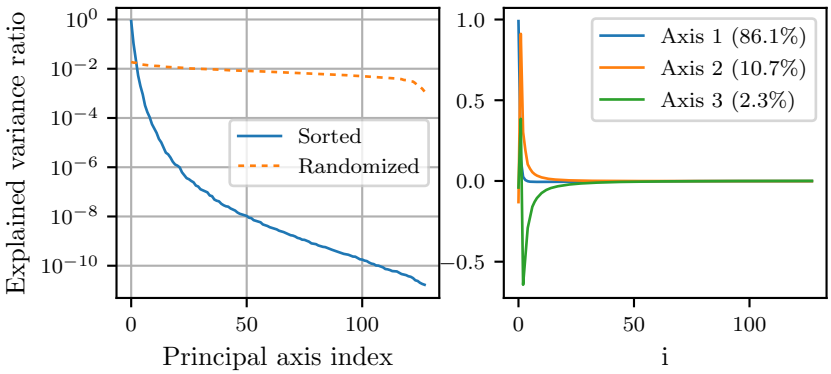


Figure 3.7: (Left) Explained variance per principal component of the PCA for two datasets: *sorted* corresponding to the truncated eigenstates p_is sorted by descending amplitude, *randomized* corresponding to truncated eigenstates whose indices have been randomized ($p_i \leftarrow p_{\mathcal{P}(i)}$ with $\mathcal{P}$ a permutation). (Right) First three principal axis, they are able to explain 99% of the variance of the *sorted* dataset.

Fig. 3.7 is illustrative of the relative simplicity of this dataset. The first three principal axis are able to explain 99% of the variance for the *sorted* data. The first principal component reveals that only the first three highest eigenstate probabilities (p_i with $i \leq 3$) are the most discriminative of the data for PCA. Interestingly, neural networks also put more weight in the first highest probabilities (see Sec. 3.4.6). This motivates us to quantify how much characteristics of the eigenstates like the highest probability (p_0) or the sum of all components ($\sum_i p_i$, not normalized to 1 anymore with our formatting) are *a priori* discriminative of our data. To do so, we calculate the variance of the dataset in the direction of these features. As a result, the highest probability (p_0) is responsible for 84.1% of the variance of the data (we did a projection onto the vector $(1, 0, \ldots, 0)$) which almost corresponds to the 85.6% of the first principal axis. The "norm" of the input vector accounts for only 1% of the total variance of the dataset (projected onto the vector $(1, 1, \ldots, 1)/\sqrt{128}$).

Listing the probabilities by descending order simplifies considerably the dataset as compared to the PCA results on the *randomized* dataset. This can be explained by the fact that with the sorted convention MBL eigenstates are all located near the vertex $(1, 0, 0, \ldots, 0)$ of the hypercube $[0, 1]^{\otimes N_c}$ where the data live. The effect of randomizing the basis states indices spread the

MBL samples across the whole hypercube, shattering them over all vertices. As a consequence, the variance distributes equally over almost all the principal components which reflects the greater complexity of this formatting as compared to the sorted input. Randomization can be seen as data augmentation since the ordering of probabilities is physically irrelevant, more problematically sorting probabilities gives artificial structure to the data that a ML analysis could exploit. In practice, we found that randomization make the training of neural networks (see next sections) harder, even impossible in some cases, which explains why we chose to the sorted convention.

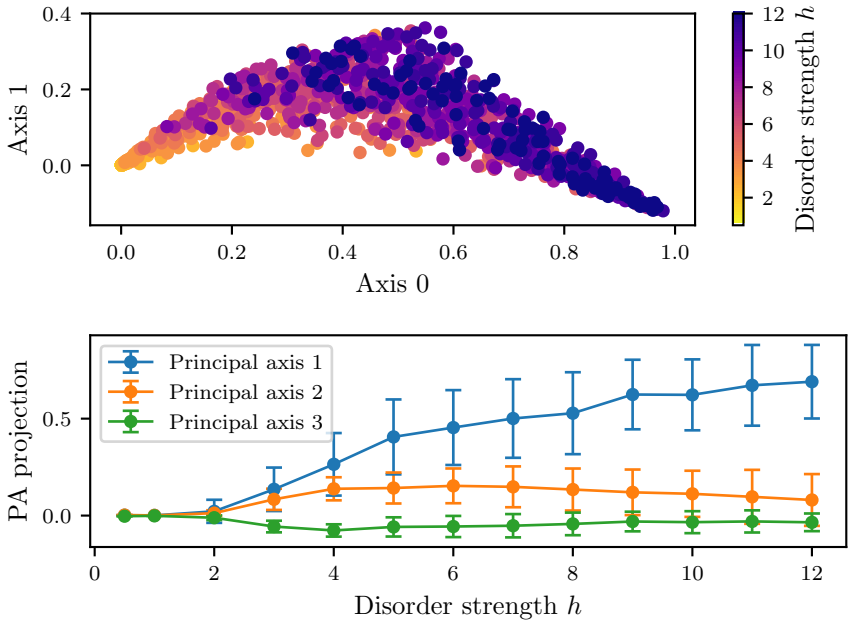


Figure 3.8: (Top) Representation of the dataset in the direction of the two first principal components, each dot corresponds to an eigenstate. The color shading indicates the disorder strength going from a fully ETH ($h = 0.5$) regime to a fully MBL ($h = 12$) regime. (Bottom) Mean of the projections of the dataset onto the three first principal components as a function of disorder strength h, the average is done on 100 different disorder realizations. The error bars correspond to one standard deviation.

Fig. 3.8 allows to visualize the PCA results in two equivalent ways. In the top panel, the data do not seem to exhibit clear clustering corresponding to different phases but rather forms a continuum going from ETH ($h \leq 1$ in yellow) data points localized on the left of the figure to fully MBL ($h \geq 10$ in blue) data points. One can notice a broader distribution of MBL samples on the first axis from one realization to the other. This is natural to expect since MBL eigenstates largely depend on the specific random fields hence a larger variance, unlike ETH eigenstates whose properties are determined uniquely by their energy. The bottom panel of Fig. 3.8 further shows the larger variance for MBL data than for ETH data, moreover it highlights the capacity of PCA to distinguish fully ETH or MBL regimes but also its limits regarding a precise determination of a finite-size crossover. Considering $h = 12$ as MBL, one can see there are MBL samples extending down to $h = 4$ where the projection on the first

component (blue curve) is inside the error bars of the $h = 12$ data. Similarly, considering $h = 0.5$ as ETH, the ETH region extends no further than $h \leq 2$. This roughly corresponds to the conventional estimates of the cross-over at $L = 16$.

3.4.3 Neural networks as phase classifiers

As previous section brought to light, a linear method is sufficient to separate strongly ETH from strongly MBL regimes, however the crossover it uncovers is too smooth to give a sharp estimate of a (finite-size) critical disorder $h_c(L)$. As a result, we resort to nonlinear methods such as neural networks with the hope that its increased power will cure the limitations of PCA. From now on, we follow the supervised approach developed by Carrasquilla and Melko [Carrasquilla & Melko 2017] (see Sec. 3.2.2) because we are quite confident on the physics at the extreme locations of the phase diagram, i.e. a fully ETH regime for $h \leq 1$ and fully MBL regime for $h \geq 10$ for all system sizes considered.

Which architecture? The question now is which neural network architecture to choose? Following our agnostic point of view, we want the most general form of a neural network. In this sense, a CNN seems too specific to respect this criterion, moreover a CNN is designed to exploit locality and translation invariance, both properties not seen by our data. As a result, we chose fully-connected feed-forward neural networks. We found that using a shallow architecture (as the one represented in Fig. 3.9) was able to fulfill many of our constraints. First, interpretability by direct inspection of the NN weights is feasible thanks to a rather small number of parameters. Second, we checked that such number of hidden neurons is optimal regarding performance on the task and variability from one NN instance to the other.

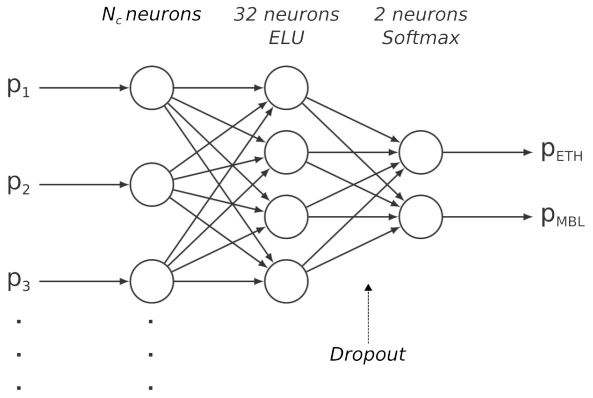


Figure 3.9: Neural network architecture used in the following. Each eigenstate is fed to the network as a vector of size N_c and the neural network outputs a number p_{MBL} in $[0, 1]$ (with $p_{\mathrm{MBL}} + p_{\mathrm{ETH}} = 1$) that is interpreted as the MBL classification confidence of the NN.

There are other advantages offered by shallow architectures. First, as seen in previous chapter, a low capacity NN needs less data to train well which can be advantageous in particular in our case since data is (exponentially) costly to generate. More crucially, it was shown in [Zhang *et al.* 2017] that state-of-the-art deep networks are able to fit randomly labelled data. They were also able to construct a two-layer neural network with ReLU activation functions and

$2n + d$ weights that could exactly memorize a dataset of n samples of dimension d. Obviously, complete memorization of the dataset is dramatic for our task since the classification would loose its ability to generalize meaningfully to the transition region. By keeping our NN small, we therefore protect ourselves from the possibility of memorization[3].

Possible bias? As shown in Fig.3.9, we applied dropout (see Sec. 2.3.5 of chapter 2) between the hidden layer and the output neuron and used ELU activation functions [Clevert *et al.* 2016]. Dropout is efficient to reduce correlations between hidden neurons since dropping connections from one training step to the other forces neurons to learn features robust and independent from the activation of other parts of the NN that could be switched off during training. Regarding the activation function, we give details in appendix A of our paper [Théveniaut & Alet 2019] about the phenomenon of dead neurons (always outputting 0) that can occur with ReLU activation functions, we found that ELU activation functions solved this issue.

NNs are universal approximators if we allow for an infinite number of units, however NNs with a finite number of parameters only span a subspace of the space of continuous function and in principle, the unknown order parameter of the transition could lie outside of this ensemble of functions. Consequently, our approach is inherently biased by the use of NNs of constant finite size. However, by monitoring the influence of parameters such as NN depth (number of layers) or width (number of neurons per layer), we could also in principle mitigate this bias.

Physical observables as neural networks. It is interesting to notice that many relevant observables such as IPRs or participation entropies can be written as relatively simple neural networks. Indeed, Fig. 3.10 shows the corresponding neural network that allows to compute such physical quantities: an operation f is applied element-wise to the input vector followed by a fully-connected layer that sum up all the neurons and give the observable. This structure resembles a CNN where an operation is performed identically on portions of the input vector (here reduced to just one value) and then a max-pooling operation reduces the dimension (here to 1). We checked that the function f in the cases of IPR ($f(x) = x^2$) or $q = 1$ participation entropy ($f(x) = -x\log(x)$) can be accurately approximated by a fully-connected neural network having a few tens of hidden neurons. This reassures us that standard shallow neural networks are expressive enough to represent usual physical observables.

Model selection. As the PCA highlighted, distinguishing strongly ETH from MBL is relatively easy and in practice, all considered neural networks achieve 100% accuracy on training and test sets. Given that the assumption is that NNs will learn features in the training dataset that are relevant over the whole phase diagram, namely over the transition region, we need another way to discriminate NNs performance.

One possibility is to ensure that the learned model achieves low bias and low variance. On the one hand, we argue that bias is low having checked that increasing the number of hidden neurons does not change the predictions, rather increasing variance. On the other hand, variance is kept small by choosing a relatively small number (32) of hidden neurons. Moreover, we can track the variance using cross-validation, i.e. obtaining multiple training instances from random

[3] In our case is $N_c \times N_{\text{hidden}} + N_{\text{hidden}} + 2N_{\text{hidden}}$ which is 8288 for the largest considered $N_c = 256$ and $N_{\text{hidden}} = 32$, to be compared to $2n + d = 400256$ for our training datasets of $n = 200000$ eigenstates ($d = N_c$)

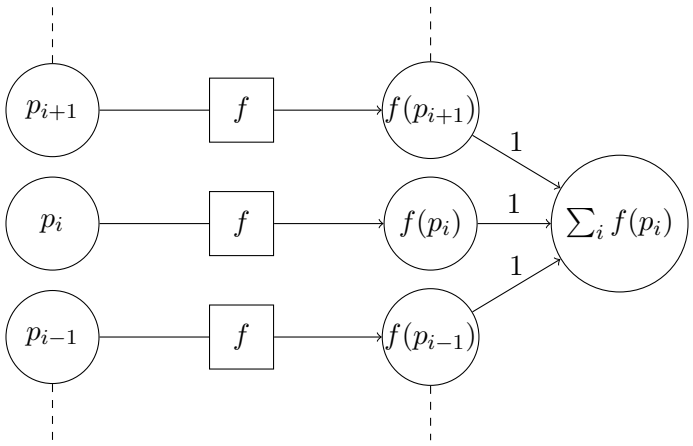


Figure 3.10: Schematic representation of a neural network computing the IPR (if $f(x) = x^2$) or the $q = 1$ participation entropy (if $f(x) = -x\log(x)$) from the vector of probabilities p_i of an eigenstate.

initialization of the NN weights and random partitioning of the training datasets (as we leave aside a fraction of the data in a separate test set). In most cases, we observe a low and stable (during training) variance with a learning rate empirically chosen at $\alpha = 0.01$, batch size of 1000 samples and number of epochs of 5. The variance gets problematic when an adversarial component is added and further comments are provided in the corresponding Sec. 3.4.8.

3.4.4 Results: Neural-network output analysis

In this section, we trained a single neural network on a dataset consisting of eigenstates obtained at $h = 0.25$ (resp. $h = 12.0$) for ETH (resp. MBL)-labelled samples for $L = 18$. The next paragraphs highlight several interesting features of the neural network predictions when applied to eigenstates over the whole parameter regime, in particular regarding (i) the typical output values of a neural-network, (ii) the variability of the predictions with respect to different training runs (iii) different disorder realizations or (iv) between eigenstates of the same realization.

Output of a typical neural network. Thanks to the use of softmax activation functions for the output layer, each eigenstate fed to the NN will produce a number between 0 and 1 that is the confidence of the NN to classify it as ETH (0) or MBL (1). Fig. 3.11 shows the typical distribution of a NN output, the distribution is unimodal for almost all disorder strengths h with a very low variance around 0 (1) in the ETH (MBL) phase, except at the transition where there is coexistence of two sharp modes (we checked that this is the case for all system sizes considered). This was not implemented as a constraint in the training of the NNs, but it turns out that the NNs output are all very close to 0 or 1, therefore to a very good approximation, *the NN output can be considered binary*, each sample being clearly identified as being either ETH or MBL (we will find an explanation for that in Sec. 3.4.6). Contrary to PCA, this already shows that NNs are able to detect sharp transitions only given samples at the extreme part of the phase diagram.

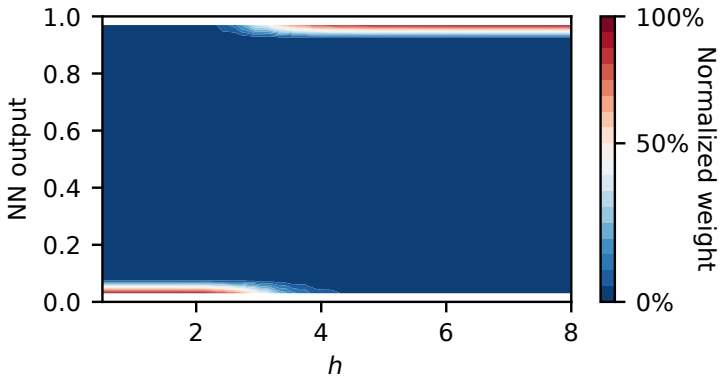


Figure 3.11: Color histogram of the output of a typical neural network trained with $L = 18$ data evaluated on 300 disorder realizations for each disorder strength.

This motivates the choice of considering the fraction f of MBL samples as a good quantity to faithfully describe the output of the neural network. The fraction f of MBL-classified samples is defined in the following way:

$$f = \frac{1}{N_\theta N_r N_i} \sum_{\theta,r,i} \Theta\left(y_{\theta,r,i} - \frac{1}{2}\right) \tag{3.8}$$

where $y_{\theta,r,i}$ denotes the output of the neural-network θ (corresponding to MBL output probability) when fed with eigenstate i from disorder realization r, Θ is the Heaviside step function, N_θ the number of trained neural networks, N_r the number of disorder realizations per disorder and N_i the number of eigenstates per disorder sample. This quantity depends on the disorder strength $f(h)$ as will be shown in Figs. 3.17 and Figs. 3.22.

We define *the finite-size critical disorder $h_c(L)$ as the disorder strength at which half of the samples are classified as MBL, meaning $f(h_c(L)) = 0.5$*. This choice is motivated by our agnostic criterion which restrain ourselves from using the fact that the critical point of model (3.7) is known to rather have MBL properties [Thiery *et al.* 2018]. Exploiting this fact would give rise to an alternative definition of $h_c(L) = \underset{h}{\text{argmin}}\{f(h) = 1\}$.

Variance from one training instance to another. Due to random initialization of the neural-network parameters and the use of stochastic gradient descent, training can in principle converge to different local minima. To quantify this effect, we pick one eigenstate per disorder realization and compute its average classification over 50 training runs denoted by $\overline{y}_r$. We do the same for the 250 different other disorder realizations and the result is showed in Fig. 3.12 as a function of h.

Fig. 3.12 reveals that there is almost no variance from run to another: all NN classify the same eigenstate almost identically (data is either clearly red or clearly blue). As we show in appendix B of our paper [Théveniaut & Alet 2019], this picture does not hold in one of the training setups we will consider later in Sec. 3.4.8 where larger fluctuations between neural networks can be seen.

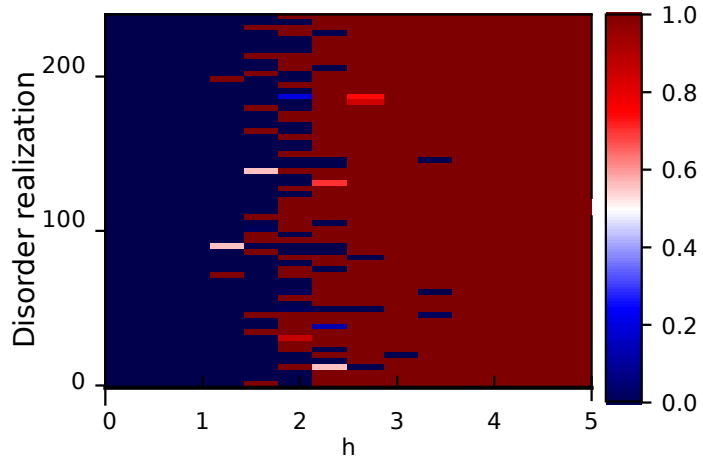


Figure 3.12: Fraction of MBL-classified eigenstates (color) over 50 training instances ($\overline{y}_r$ as defined in the main text) as a function of disorder strength (x-axis) and realization number (y-axis) when training and prediction are done on $L = 18$ data. One eigenstate per disorder realization is fed to every NN.

Eigenstate-to-eigenstate, sample-to-sample variance First, we consider variations of classification from one disorder realization to another. For a given neural network θ, we study the average classification of individual eigenstates sharing the same disorder realization r denoted as $\overline{y}_\theta(r)$. Fig. 3.13 shows an histogram of $\overline{y}_\theta(r)$ for a typical NN θ for 250 disorder realizations. We have checked that this picture is stable for all training instances and for any of the considered training setups.

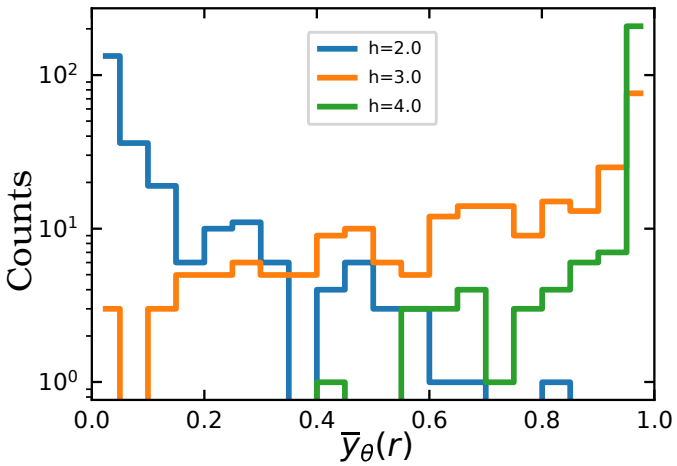


Figure 3.13: Histogram of $\overline{y}_\theta(r)$ obtained on 250 disorder realizations. Training and prediction are obtained from $L = 18$ data.

For disorder strengths slightly lower (resp. higher) i.e. at $h = 2.0$ (resp. $h = 4.0$) than the crossover point (here around $h = 3.0$ for $L = 18$), the distribution is peaked around 0 (resp. 1) meaning that almost all eigenstates from any disorder realizations are classified as ETH (resp.

MBL). More interestingly, for $h = 3.0$ the NN detects both ETH and MBL eigenstates within the same disorder realization. The fact that, close to the transition, the network predicts both ETH and MBL eigenstates in the same disorder realization at the same energy density is reminiscent of what was observed in [Yu *et al.* 2016], where a bimodal distribution of entanglement entropy was observed also at the individual disorder realization level close to the transition. This motivated us to look at the correlation between the prediction of each eigenstate and its entanglement entropy. Our analysis in appendix B of our paper [Théveniaut & Alet 2019] clearly shows that eigenstates with low (high) entanglement are systematically classified as MBL (ETH), which also implicitly indicates that our input formatting is not too destructive.

Error bars. The discussion above allows to simplify the analysis of the NN output by: (i) describing the NN output as binary ($\{0, 1\}$), (ii) neglecting the variance coming from the stochasticity of training. Moreover, as eigenvectors of the same disorder realization are correlated, we chose to bin quantities over all eigenstates of the same realization and all neural networks, and then compute the standard error over these bin averages in order not to underestimate error bars. This is similar to what was done in the conventional analysis of [Luitz *et al.* 2015]. The only remaining variability originates from disorder realizations, as showed by the error bars in the forecoming plots.

3.4.5 Neural-network setups

To identify the phases and obtain critical exponents of the transition, we use the finite-size scaling method explained in Sec. 3.1.2.

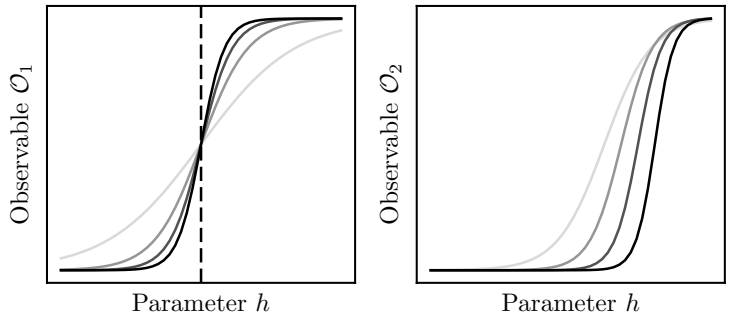


Figure 3.14: Two typical observables' behaviours arising in finite-size simulations (darker grey means larger system size): (left) a unique crossing point is visible which unambiguously locates the size-invariant critical point, (right) a "drift" of the finite-size curves prevents from identifying the critical point.

First, let us observe Fig. 3.14 where two observables and their finite-size trends are showed. One the one hand, the left plot exemplifies a situation where a unique crossing point is visible which allows to identify the critical point directly. This was the situation encountered in [Carrasquilla & Melko 2017] for the neural-network output for the Ising model, or for the MBL transition in [Huembeli *et al.* 2019]. On the other hand, when curves rather drift and do not cross, one can alternatively try to define a finite-size pseudo critical point $h_c(L)$ (with some criterion) and naturally assume a finite-size relation $h_c(L) - h_c(\infty) \sim L^{-1/\nu}$. This was for instance

used in Refs. [Li *et al.* 2019, Zhang *et al.* 2019a]. In our case, we find (see Figs. 3.17, 3.22) that the latter situation applies (no crossing of curves) and thus assume the second scaling form. The definition of $h_c(L)$ was given in Sec. 3.4.4.

As a disclaimer, the use of approximate ML techniques in combination with such a sophisticated procedure like finite-size scaling seems uncertain. Indeed, there is no constraint on the ML procedure to uncover a classification rule that corresponds to a physical order parameter or physically interpretable quantity which correctly captures critical phenomena. Due to stochasticity in the training procedure, all other things being equal, two independent trainings could for instance converge to a NN approximating an order parameter in one run and converge to a NN approximating its Binder cumulant for the other run (or any combination thereof), which are known to exhibit different critical behavior and finite size effects. This is especially problematic when the finite-size scaling is attempted from the aggregation of different training instances. As a result, our work consisted in designing setups that would mitigate as much as possible these harmful effects.

We investigated the two following setups to perform finite-size scaling:

- **Single-size setup.** An ensemble of NNs are separately trained on different systems sizes, a finite-size critical disorder $h_c(L)$ can be deduced from the output of each NN, finally we attempt to obtain $h_c(\infty)$ and ν by fitting these $h_c(L)$ to the scaling form shown in the figure below. The results will be presented in Sec.3.4.6.

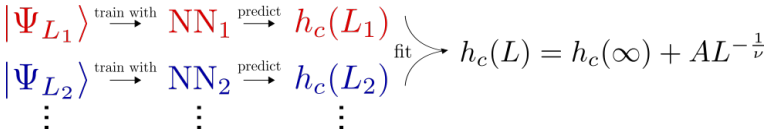


Figure 3.15: Single-size training setup.

- **Multiple-size setup.** One NN is trained on a dataset containing data from multiple system sizes at once. The results are discussed in Sec.3.4.7. We will consider an augmented version of this setup in Sec.3.4.8 to achieve better generalization across system sizes.

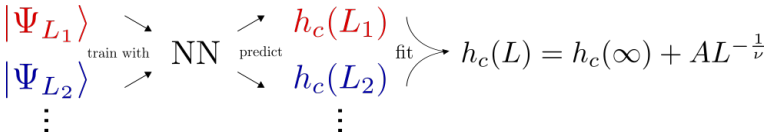


Figure 3.16: Multiple size training setup.

3.4.6 Results: Single system size training

We present in this section the results obtained with the single-size training setup. We study the predictions of five neural networks trained on data respectively from $L = 14, 16, 18, 20, 22$. Apart from the training dataset, all hyperparameters (learning rate, batch size, number of epochs, etc..) and NN architecture (number of hidden neurons, etc..) are fixed. The training dataset consists of eigenstates obtained at $h = 0.25$ (resp. $h = 12.0$) for ETH (resp. MBL)-labelled samples for all system sizes.

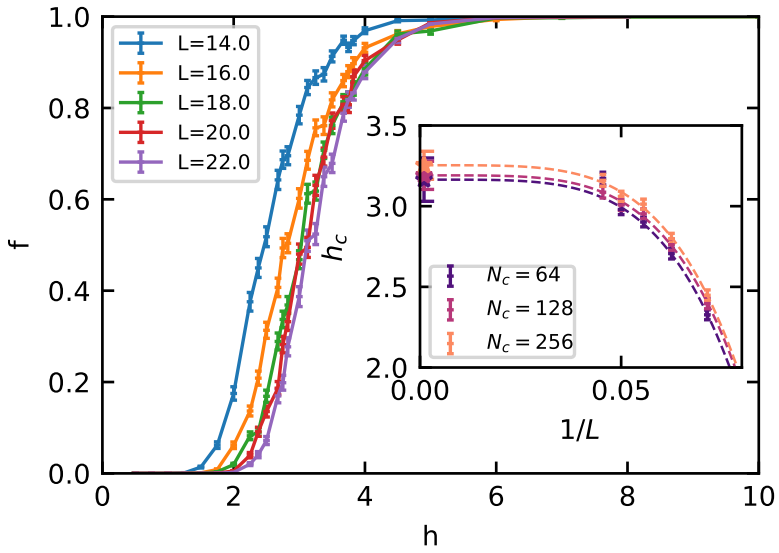


Figure 3.17: Fraction of MBL-classified samples as a function of disorder strength for NN trained on a given system size L. Predictions are averaged over 250 disorder realizations per disorder (with 100 eigenstates per realization) and 50 training instances. Truncation order is $N_c = 256$. The error bars indicate the statistical error due to sampling disorder realizations. Inset: finite-size scaling analysis of $h_c(L)$ defined as $f(h_c(L)) = 0.5$ for different truncations N_c. The error bars on the final estimates come from the fitting procedure.

Several features can be distinguished in Fig. 3.17: one is the existence of a fully ETH region (where all samples are classified as ETH) that extends from $h = 0$ to $h = 2$ and a fully MBL region starting from $h = 6$ for all system sizes. Another distinct feature is the hierarchy of the curves depending on the system size L, i.e. the crossover from ETH to MBL happens for higher disorder as L is increased. This behavior is in agreement with many other observables (such as spectral statistics, entanglement variance, dynamical spin fraction) used in the standard analysis of this system [Luitz *et al.* 2015], which also display regions where ETH and MBL are clearly well identified, and a crossover region with a right-shift (*i.e.* towards larger disorder) of the finite-size estimate of the transition point with system sizes (similar to the right panel of Fig. 3.14).

Interpretation of the neural network — The most straightforward way to understand what a NN learnt is to directly look at its weights. The neural network structure we used (showed in Fig. 3.9) encodes the following function:

$$f(\{p_i\})_n = \text{softmax}\left[\left(\sum_j W^{(2)}_{nj}\text{ELU}\left(\sum_i W^{(1)}_{ji}p_i + b^{(1)}_j\right) + b^{(2)}_n\right)\right]_n \tag{3.9}$$

where $W^{(1)}$ and $W^{(2)}$ are matrices of dimension respectively $(32, N_c)$ and $(2, 32)$, $b^{(1)}$ and $b^{(2)}$ are vectors of dimension 32 and 2, respectively. Note that softmax is a function that takes a

vector $\mathbf{z}$ and outputs $\text{softmax}(\mathbf{z})_n = e^{z_n}/\sum_n e^{z_n}$. These variables constitute the parameters of the neural network, their values obtained after training are shown in Fig. 3.18.

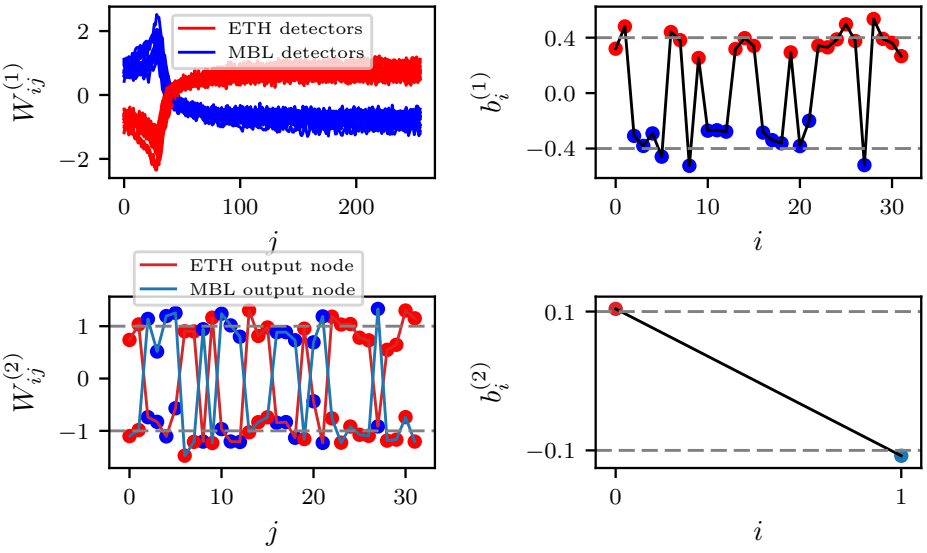


Figure 3.18: Values of the internal weights of a typical NN trained on $L = 18$ data with $N_c = 256$. (Top left) The set of 32 weights $W_{ij}^{(1)}$ ($i \in [1, 32]$) are represented as a function of the input neuron index ($j \in [0; N_c]$) and coloured according to the category we assigned them: being dubbed MBL detectors (blue) or ETH detectors (red). (Top right) The biases $b_i^{(1)}$ of each hidden node i are shown, where the color of the dots tells to which category neuron i belongs to, ETH (red) or MBL (blue) detectors. (Bottom left) The weights $W_{0j}^{(2)}$ (resp. $W_{1j}^{(2)}$) outputting the ETH-classification (resp. MBL-classification) confidence is shown in light red (resp. light blue) as a function of j the hidden node index where we added a blue (red) dot to indicate the category of the hidden node.

The top left plot of Fig. 3.18 highlights the existence of two categories of hidden neurons: (i) (resp. (ii)) half of the neurons weigh positively (resp. negatively) the largest probabilities p_i (corresponding to the smallest input indices) until input index $i \simeq 40$ then the next inputs are weighed negatively (resp. positively). Let us denote category (i) MBL detectors and category (ii) ETH features, this denomination will be clearer later. We denote the weights of MBL detectors as W_i^{MBL}. The biases of the hidden nodes ($b^{(1)}$) as well as the weights coming from the hidden layer to the output layer ($W^{(2)}$) have a behaviour that can be directly deduced from the category of the hidden node (whether it is a MBL detector or an ETH detector) it is linked to. In particular, Fig. 3.18 reveals the following relations (following Eq. 3.9 conventions):

- if hidden node j is a MBL detector, then $W_{j,i}^{(1)} \simeq W_i^{\text{MBL}}$, $b_j^{(1)} \simeq -0.4$, $W_{0j}^{(2)} \simeq -1$ and $W_{1j}^{(2)} \simeq 1$.

- if hidden node j is an ETH detector, then $W_{j,i}^{(1)} \simeq -W_i^{\text{MBL}}$, $b_j^{(1)} \simeq 0.4$, $W_{0j}^{(2)} \simeq 1$ and $W_{1j}^{(2)} \simeq -1$.

The relations above allows to obtain a simplified expression of the argument of softmax in Eq. 3.9:

$$\pm \sum_{j\in \mathrm{ETH}} \mathrm{ELU}\left(-\sum_i W_i^{\mathrm{MBL}} p_i + 0.4\right) \mp \sum_{j\in \mathrm{MBL}} \mathrm{ELU}\left(\sum_i W_i^{\mathrm{MBL}} p_i - 0.4\right) \pm 0.1 \tag{3.10}$$

Given that the softmax activation function is applied to 2-component vector such that $\mathbf{z} = (z_0, -z_0)$ (as showed in Eq. 3.10), this gives $\mathrm{softmax}(\mathbf{z})_k = \mathrm{sigmoid}(2z_0)$. Moreover, there is on average as many ETH detectors as MBL detectors, which further simplifies Eq. 3.10 as follows:

$$f_{0,\mathrm{ETH}}(\{p_i\}) = \mathrm{sigmoid}\left[32g\left(-\sum_i W_i^{\mathrm{MBL}} p_i + 0.4\right) + 0.2\right] \tag{3.11}$$

$$f_{1,\mathrm{MBL}}(\{p_i\}) = \mathrm{sigmoid}\left[-32g\left(-\sum_i W_i^{\mathrm{MBL}} p_i + 0.4\right) - 0.2\right] \tag{3.12}$$

with $g(x) = ELU(x) - ELU(-x)$. It is possible to identify, to a good approximation, the function $x \mapsto \mathrm{sigmoid}\,[32g(x) + 0.2]$ by the Heaviside step function, consequently we can summarize the action of the neural networks as follows:

1. Given eigenstate $\mathbf{p} = (p_i)$, compute the scalar product $C = \mathbf{W}^{\mathrm{MBL}} \cdot \mathbf{p}$
2. Classify eigenstate $\{p_i\}$ as being ETH if $C < 0.4$, MBL otherwise.

(3.13)

This shows that the action of the neural network essentially boils down to a linear operation on the input data followed by a thresholding operation. Compared to PCA, the sharpness of the neural-network predictions directly comes from the application of the sharp non-linear threshold. Fig. 3.19 shows a histogram of $\mathbf{W}^{\mathrm{MBL}} \cdot \mathbf{p}$.

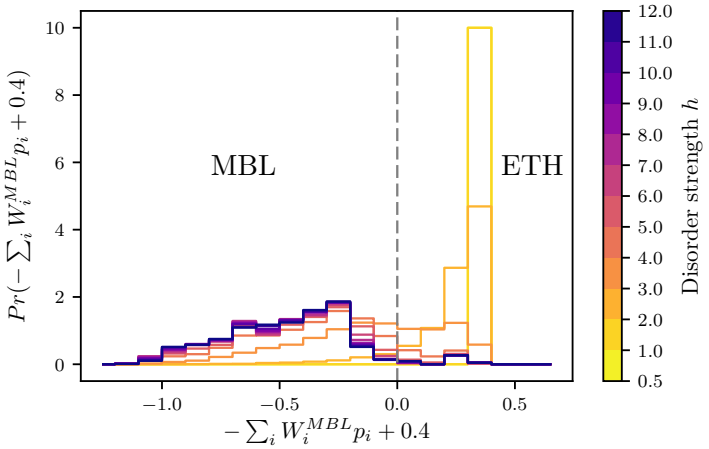


Figure 3.19: Histogram of the activations of the first hidden layer $-\sum_i W_i^{\mathrm{MBL}} p_i + 0.4$ computed over 32 hidden neurons of a neural-network trained at $L = 16$ and 100 disorder realizations per disorder strength (different colors). The dotted line is the classification threshold.

Alternatively, the shape of W_i^{MBL} (see top left plot in Fig. 3.18) points towards the relevance of the participation entropies S_q^P for high values of q (as the largest p_i are more weighted by the

NN), as a feature to classify the phases and detect the transition. The particular relevance of the IPR ($q = 2$) was also noted in the support vector machine analysis of a MBL transition in [Zhang *et al.* 2019a].

Influence of truncation order N_c — Fig. 3.20 shows the averaged weights W_i^{MBL} after training is done with different truncation orders N_c. We checked that all the other parameters ($b^{(1)}$, $b^{(2)}$, $W^{(2)}$) of the NN roughly take the same values as shown in Fig.3.18 irrespective of the system size L or the training instance. These results show that the "order parameter" uncovered by the NN is consistent as we increase the size of the input sample, i.e. as we provide more eigenvalue probabilities p_i. More formally, we have: $(W_i^{\mathrm{MBL},N_c=128})_{i\in[[1,64]]} \simeq W_i^{\mathrm{MBL},N_c=64}$ and so on.

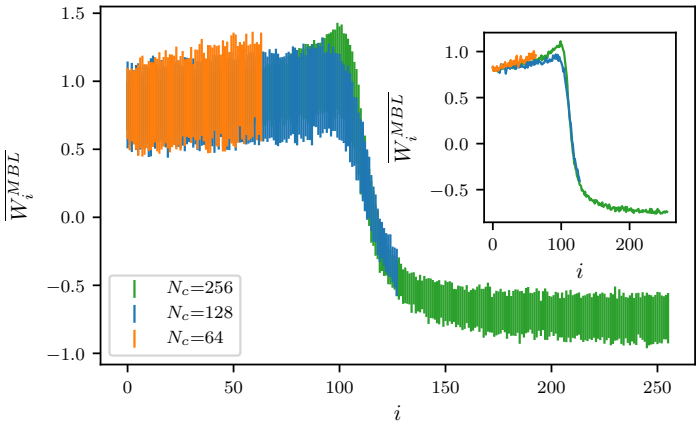


Figure 3.20: Weights of the first hidden layer W_i^{MBL} for different truncation order $N_c = 64, 128, 256$ for a NN trained on $L = 22$ data, averaged over the 32 hidden neurons of the NN. The inset shows the average weights without error bars for clarity purposes.

Weight scaling with system size L — We notice that the existence of MBL/ETH detectors persists for different system sizes, moreover the values of the biases $b^{(1)}$, $b^{(2)}$ and weights of the second layer $W^{(2)}$ take almost the same values. However, there is variability for the weights $W^{(1)}$ as can be seen in the left plot of Fig. 3.21.

Quite remarkably, despite any constraint the neural networks trained on different system sizes learn the same classification rule modulo a certain rescaling with system size L. The separation between positive and negative weighing of the input data is pushed further (towards larger input index number) as L increases. This is shown in the right plot of Fig.3.21 where the index number of the maximum of W_i^{MBL} follows a scaling law of the form $\sim |\mathcal{N}(\mathcal{L})|^D$ where $|\mathcal{N}(\mathcal{L})|$ is the size of the Hilbert space for system size L considered. We have not found a precise explanation for this scaling but we can make the following speculation: it suggests that the neural networks detect eigenstates that have a multifractal dimension $D < 1$. We know that eigenstates in the MBL regime have $D < 1$ and those in the ETH regime $D = 1$ [Macé *et al.* 2019]. Since the NN was trained on both MBL and ETH eigenstates, we can perhaps interpret D as an average over all fractal dimensions, which would constitute an upper (resp. lower) bound for the fractal

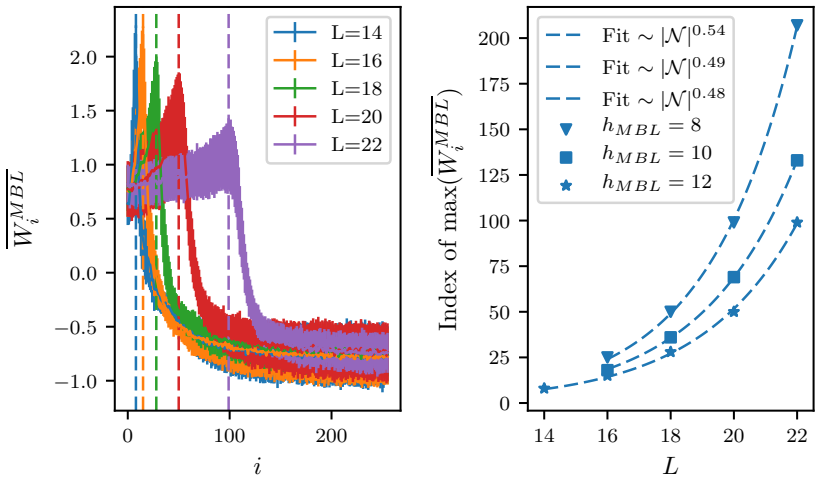


Figure 3.21: (Left) Weights of the first hidden layer W_i^{MBL} for different NNs trained separately at system sizes from $L = 14$ to $L = 22$ with $N_c = 256$ and dataset consisting of ETH-labelled eigenstates from $h_{\mathrm{ETH}} = 0.25$ and MBL-labelled eigenstates from $h_{\mathrm{MBL}} = 12$. The vertical dashed lines indicate the locations of the maximum of W_i^{MBL} (stars in right panel). (Right) Locations of the maxima of W_i^{MBL} are plotted as a function of L for different training datasets having different $h_{\mathrm{MBL}} = 8$, $h_{\mathrm{MBL}} = 10$, $h_{\mathrm{MBL}} = 12$. A scaling law of the form $|\mathcal{N}|^D$ is fitted for each h_{MBL} with $|\mathcal{N}|$ the size of the corresponding Hilbert space.

dimension of a MBL (resp. ETH) eigenstate. Moreover, the "fractal" dimension depends on the training dataset and as the disorder strength used for training h_{MBL} is decreased, the fractal dimension slightly increases which seems to support our hypothesis.

Finite-size scaling — Previous paragraph showed that neural networks trained on different system sizes L encode the same observable but stay different because they have to adapt to the scaling with L of the input values with a shared constant architecture. Given the scalability of these results, it seems reasonable to perform a finite-size scaling analysis. The finite-size scaling results for different truncation order $N_c = 64, 128, 256$ are summarized in Table 3.1. In practice, we approximate the fraction f by a cubic polynomial around the putative $h_c(L)$ fitted in the interval $[h_c(L) - w; h_c(L) + w]$ with $w = 0.6$ (giving the smallest error bars).

Truncation	h_c	ν	χ^2/dof
$N_c = 64$	3.16 ± 0.13	0.23 ± 0.07	0.03
$N_c = 128$	3.19 ± 0.09	0.22 ± 0.06	0.13
$N_c = 256$	3.25 ± 0.09	0.23 ± 0.05	0.32

Table 3.1: Finite-size scaling results with single-size training, as a function of truncation order N_c. The scaling ansatz is $h_c(L) = h_c(\infty) + AL^{-\frac{1}{\nu}}$.

The scaling procedure leads to a critical disorder value $h_c \simeq 3.2$ that is lower than the usual estimate around $h_c \simeq 3.7$ [Luitz *et al.* 2015], and extremely small values of $\nu \simeq 0.22$, which appears unreasonable. The underestimation of the critical disorder seemingly comes from the

truncation preprocessing step, indeed h_c increases as N_c increases. Note that we needed to take aside $L = 22$ data for $N_c = 64$ (otherwise having $\chi^2/\text{dof} = 0.95$) : the number of truncated probabilities is too small to get a meaningful result for the largest size (see Fig. 3.20).

Discussion — Even though our results show consistent behaviours with respect to N_c or L, this setup could allow for the unfortunate possibility that a NN trained on a given L learns (*i.e.* reproduces the features of) a certain physical observable different from the one learned for a NN trained at a different L. Indeed, learning a certain classification model depends for instance on the NN capacity (number of layers / hidden neurons) relative to the complexity of the training dataset (that varies from one system size to another). It has already been noticed that non-universal size-dependent features were indeed captured by neural networks [Beach *et al.* 2018]. Even more dramatically, [Ponte & Melko 2017] showed that different physical observables are learned depending on the amount of regularization, though this happened with support vector machines.

In the next section, we use the multi-size setup that attempts to solve the latter issue by allowing for the learning of size-invariant features. In addition to that, it will in principle permit meaningful transfer learning like detecting the transition on L_1 data from a model trained on $L_2 \neq L_1$ data.

3.4.7 Results: Multiple system size training

The chosen formatting of input data (Sec. 3.4.1) with fixed size allows us to use one unique NN to treat data from different system sizes on equal footing. We hope that this will help the neural network to capture size-invariant features, i.e. features in the thermodynamic limit, in particular close to criticality. We investigate in the following what a neural network trained on a dataset containing system sizes $L = 16, 18, 20, 22$ all at once can learn and compare the results to the previous analysis (we refrain from using $L = 24$ data as not enough samples are available for training). To do so, we need to work at constant truncation order N_c whatever system size is picked for training. The dataset has the same size as in the previous section, taking one fourth of samples from $L = 16$ data, one fourth from $L = 18$ and so on.

Fig. 3.22 shows the fraction of MBL-classified samples as defined in Sec.3.4.4 and displays similarities with Fig. 3.17 regarding the existence of fully-ETH and fully-MBL regimes located at the same regions. Nevertheless a striking asymmetry from single-size training appears: a broadening of the curves in the crossover region. Fig. 3.22 also features non-trivial transfer learning: a neural network trained on $L = 16, 18, 20, 22$ is asked to classify samples from system sizes $L = 14$ and $L = 24$ for which it has never seen any samples before. This highlights one advantage of this multi-size training setup, namely its reduced computational cost. It is indeed reduced by a factor proportional to the number of considered system sizes and number of retrainings, which can represent a huge saving in computation time.

Interpretation of the neural networks — To understand the differences between the predictions of the single-size setup (see Fig. 3.4.6) and the multi-size setup (see Fig. 3.4.7), we performed in [Théveniaut & Alet 2019] a PCA on the weights $W^{(1)}$ of the first hidden layer (see Eq. 3.9). More precisely, we form a matrix by stacking the 32 columns of $W^{(1)}$ (i.e. $(W_{ji})_{i=1\ldots N_c}$) as there are 32 hidden nodes for 5 different training instances, the matrix is then of dimension

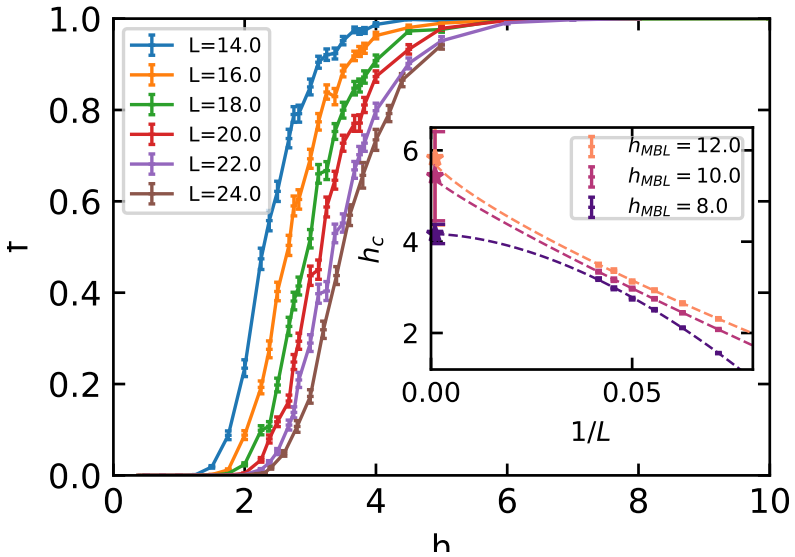


Figure 3.22: Fraction of MBL-classified samples as a function of disorder strength for a NN trained on multiple system sizes all at once and evaluated on different system sizes. Predictions are averaged over 250 disorder realizations per disorder (with 100 eigenstates per realization) and 50 training instances. Truncation order is $N_c = 256$. The error bars indicate the statistical error due to sampling disorder realizations. Inset: finite-size scaling analysis using ETH-labeled data at $h_{\mathrm{ETH}} = 0.25$ and MBL-labeled data at $h_{\mathrm{MBL}} = 8.0, 10.0, 12.0$. The error bars on the final estimates come from the fitting procedure.

$(32 \times 5, N_c)$. Fig. 3.23 shows the weights $(W_{ji})_{i=1\ldots N_c}$ after a projection onto the two principal axis.

We find this analysis less informative than a direct inspection of the weights as done for the single-size setup, nevertheless Fig. 3.23 has the merit of providing a simple picture of the situation. The weights split up into two symmetric categories by a sign change, this corresponds to the MBL and ETH detectors revealed previously, visible here through the PCA representation. It also confirms that single-size training on L-data leads to capturing L-specific features. A hierarchy appears where the weights corresponding to training at a given system size L are next to the weights for $L \pm 2$.

The weights learned in the multiple-size setup overlap the weights of the NN trained at $L = 18$ and $L = 20$, capturing an averaged model of the system sizes $L = 16, 18, 20, 22$. This shows that the NN does not actually capture size-independent features (which would manifest by a uniform distribution of weights over the L-specific subspace of weights) but rather in a weaker way, it uncovers averaged features that are shared by all the provided system sizes. To corroborate this point, we trained a NN on system sizes $L = 14, 16, 18, 20$ and we also notice the same averaging behaviour, i.e. this time the NN captured features similar to those captured for $L = 16$ and $L = 18$ trainings.

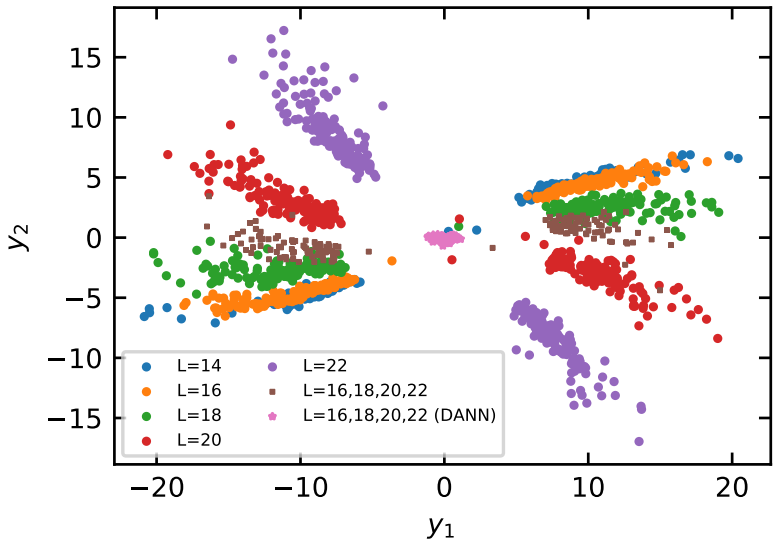


Figure 3.23: PCA representation of the weights learned after training on single system size datasets (Sec. 3.4.6), multiple system size datasets (Sec. 3.4.7), supplemented by an L-adversarial component (Sec. 3.4.8). Each dot is a weight projected on the two principal axis of the PCA analysis (which accounts for 90% of the total variance). 5 training instances are included for each training case.

Finite-size scaling — We perform a finite-size scaling with varying training datasets which include MBL-labelled samples drawn from different disorder strengths $h_{\mathrm{MBL}} = 8, 10, 12$ while the ETH-labelled samples are all taken from $h_{\mathrm{ETH}} = 0.25$, because we noticed negligible change in the scaling for $h_{\mathrm{ETH}} = 0.5$ or $h_{\mathrm{ETH}} = 1.0$. We found that including predictions obtained by transfer learning at $L = 14$ and $L = 24$ system sizes considerably improve the results, in the sense that the fitting procedure converges with rather small error bars on h_c and ν. If $L = 14$ is taken aside, the error bars are multiplied by a factor of 4 and the fits do not converge if no transfer learning is done (performing the fit only on $L = 16, 18, 20, 22$).

Data	h_c	ν	χ^2/dof
$h_{\mathrm{MBL}} = 8.0$	4.17 ± 0.04	0.58 ± 0.01	/
$h_{\mathrm{MBL}} = 10.0$	4.93 ± 0.10	0.93 ± 0.04	/
$h_{\mathrm{MBL}} = 12.0$	5.74 ± 0.21	1.27 ± 0.09	/

Table 3.2: Finite-size scaling results with multiple-size training, for different values of the training disorder used to label the MBL phase. Truncation order is $N_c = 256$. The absence of χ values come from our specific scaling procedure, see note [76] of [Théveniaut & Alet 2019]

The finite-size scaling analysis with varying training datasets leads to a somewhat unexpected result: whereas it is generally considered that the $h > 8$ region contains only strongly MBL eigenstates with very similar physical properties, the neural networks learn nevertheless different models resulting in estimates of h_c ranging from $h_c \simeq 4$ to $h_c \simeq 6$, higher than the estimated

value, and ν ranging from 0.6 to 1.4. This phenomenon can be rationalized with the following naive argument: samples in the transition region will be classified MBL for lower disorders if the MBL-labelled samples are themselves taken from region closer to the transition, thus shifting the transition point towards lower critical disorder. As discussed earlier for the single-size setup, one can speculate that this finding actually echoes the non-universal multifractal properties of the MBL phase noticed in [Macé *et al.* 2019] and based on the same type of input data. Indeed one can associate a different multifractal dimension (decreasing with h) to every h_{MBL}: the h_{MBL}-dependence could then be viewed as the manifestation of the varying multifractality in the MBL phase. To circumvent this issue, one may for instance include samples from a range of disorder values all at once. However, we noticed that if we provide a training dataset containing MBL-samples from $h_{\mathrm{MBL}} = 8, 10, 12$, the NN tends to capture h_{MBL}-averaged features of the dataset (see next paragraph), i.e. leading to predictions similar to those of a NN trained at $h_{\mathrm{MBL}} = 10$.

Discussion — The multiple-size training setup was expected to produce more reliable predictions i.e. that would be less prone to learning L-dependent features. Nevertheless, we found the transition point greatly depends on the region of the phase diagram used for training (this was also noticed in Refs. [Ch'ng *et al.* 2017, van Nieuwenburg *et al.* 2017]). This is clearly a limitation of our setup since one would want the critical parameters to be insensitive to the location of the training data in the phase diagram.

The analysis of the weights revealed that this setup leads to the learning of an averaged model of the system sizes provided in the dataset, i.e., the neural network in the multi-size setup computes the same observable as a neural-network that we would obtain from training on a single system size $L = \frac{16+18+20+22}{4}$. The multi-size predictions are then obtained from a neural-network that does not scale with system size while the input data naturally does (the first N_c highest p_i values have a larger tail for larger system sizes) and the different broadening of the curves (see Fig. 3.16) can perhaps be interpreted as an artifact of this. A possible solution would be to normalize the input data for the multiple-size setup, which we have not attempted in [Théveniaut & Alet 2019].

In next section, we will try to circumvent these limitations by adding a constraining element in the NN architecture such that it will prevent the NN from capturing size-dependent features or size-averaged behaviors.

3.4.8 Results: System size adversarial training

The two previous sections pointed out the difficulty to fight against dataset dependence of the NN predictions. Domain-adversarial neural networks (DANN) have been introduced in [Ganin *et al.* 2016] in order to tackle domain adaptation, *i.e.* when the datasets at training and prediction time come from similar but different distributions. The goal of domain adaptation is to learn features that cannot discriminate between the training and prediction domains. This particularly fits the problem of phase classification since we would like the features captured in the extremal regions of the phase diagram during training to be generic enough to describe the intermediate parameter region during prediction. This idea was introduced for phase classification in [Huembeli *et al.* 2018] and proved successful for many types of transitions [Huembeli *et al.* 2019].

It is possible to expand these ideas and constraint a NN to learn features that are insensitive to the system sizes it has been trained on, i.e. L-invariant. To do so, we design a DANN containing two supplementary components as shown in Fig. 3.24: a system size classifier and a gradient reversal layer. The presence of the system size classifier means that every eigenstate in the training dataset will have not only a phase label but also a *size label* which simply tells from which system size the sample is from. The gradient reversal component works by leaving the input unchanged during forward-propagation and reverses the gradient by multiplying it by a negative scalar during the back-propagation. Its effect is crucial because it changes the sign of the derivatives of the feature extractor parameters with respect to the size classifier output. As a consequence, the parameters of the feature extractor will be optimized to make the task of the phase classifier as easy as possible while making that of the system size classifier as hard as possible. If the network reaches equilibrium, the selected features are the best suitable to identify which phase a sample lies in, while containing no information about which system size it emanates from.

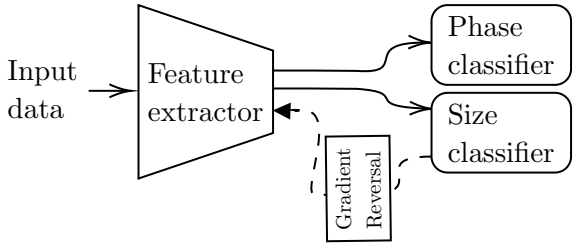


Figure 3.24: Neural network containing an adversarial component applied on the system size label.

Learning L-invariant features — The feature extractor is identical to the neural network (Fig. 3.9) used in previous sections. The system size classifier consists of 4 softmax neurons corresponding to each provided system size ($L = 16, 18, 20, 22$) and outputs what can be interpreted as the probability of a sample to be from a given system size. The loss function contains now an additional term corresponding to the size classifier:

$$\mathcal{L} = \sum_{x,\mathbf{y}^h,\mathbf{y}^L} \underbrace{\sum_{j=1}^{2} y_j^{\mathrm{h}} \log\left(f_j^{\mathrm{h}}(x)\right)}_{\text{Phase classifier loss}} + \underbrace{\sum_{j=1}^{4} y_j^{\mathrm{L}} \log\left(f_j^{\mathrm{L}}(x)\right)}_{\text{Size classifier loss}} \tag{3.14}$$

where $\mathbf{y}^{\mathrm{h}}$ (resp. $\mathbf{y}^{\mathrm{L}}$) is the two-dimensional (resp. four-dimensional) one-hot vector representing the phase label (resp. the system size label) of sample x, $\mathbf{f}^{\mathrm{h}}$ (resp. $\mathbf{f}^{\mathrm{L}}$) is the corresponding two-dimensional (resp. four-dimensional) softmax output of the phase classifier (resp. the system size classifier). Because of the adversarial component, the optimization process will keep the size classifier loss at much higher values (in practice orders of magnitude larger) than the phase classifier loss: the NN will thus be discriminative for the phase classification task and indiscriminate with respect to the shift between the $L-$data domains.

Adversarial learning is generally considered to be a hard task [Lucic *et al.* 2018], for instance non-convergence can occur with oscillations of the optimized parameters. Training is known to be

very sensitive to the hyperparameter selections since any unbalance between the two adversaries can lead to overfitting or other unwanted phenomena. In particular, we noticed that the weights of the feature extractor tended to take arbitrarily large values (increasing with training time). This has the effect of increasing the variance of the predictions from one training instance to another and may also cause overfitting. Therefore we found crucial to add a L^2 weight decay term in the cost function (3.14), in the form $\mu|W|^2$ with W being the internal parameters of the feature extractor. This regularization technique requires however a good choice of μ. After fine-tuning, we found that $\mu = 0.05$ gives good results. We checked that the finite-size scaling of previous section with the same regularization (weight decay with $\mu = 0.05$) gives same critical values with no better error bars.

Interpretation of the neural network — The PCA representation of the DANN weights in Fig. 3.23 shows that this setup allows some apparent independence of the model with respect to system size since the weights are homogeneously distributed over L-specific weights subspaces. Fig. 3.25 shows the matrix of weights (of dimension $(4, 32)$) connecting the feature extractor to the size classifier. The L-invariance property of the NN is achieved by reaching the following trivial equilibrium configuration: the output of the feature detector is multiplied by the weight vector $\mathbf{W}_L$ to the L-output of the size classifier with $L = 16, 18, 20, 22$ and Fig. 3.25 shows precisely that $\mathbf{W}_{L=16} = \mathbf{W}_{L=18} = ...$ As a result, any sample has an equal probability of belonging to any of the provided system sizes, in other words, the output of the size classifier is $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$ for all input.

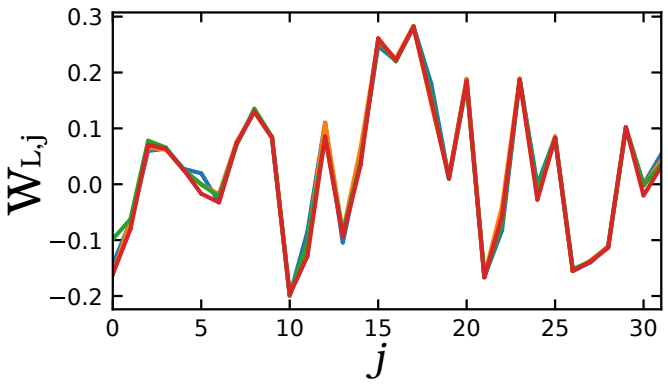


Figure 3.25: Weights connecting the feature extractor to the system-size classifier plotted against hidden layer neuron index of a training instance of a DANN trained on $L = 16, 18, 20, 22$ data for $N_c = 256$. Each color corresponds to one of the 4 size-classifier neurons.

Finite-size scaling — We notice various improvements from last section. The adversarial component helps reducing the training region dependence noticed before as well as in [Huembeli *et al.* 2019]. The critical disorder $h_c \simeq 5.5 - 6$ is still higher than the conventional estimate and $\nu \simeq 1.2$ is also (slightly) higher. We stress that these results cannot be compared to the ones obtained in [Huembeli *et al.* 2019] because the setup used there differs in several ways: whole eigenfunctions (up to size $L = 18$) and single-size training are used, and the adversar-

ial component is used differently to reduce the discrepancy between samples from the strongly ETH/MBL regions and from the intermediate region.

Training data	h_c	ν	χ^2/dof
$h_{\mathrm{MBL}} = 8.0$	5.38 ± 0.14	1.14 ± 0.06	/
$h_{\mathrm{MBL}} = 10.0$	5.75 ± 0.19	1.37 ± 0.08	/
$h_{\mathrm{MBL}} = 12.0$	6.16 ± 0.34	1.54 ± 0.15	/

Table 3.3: Finite-size scaling results using a DANN approach for multiple-size training, as a function of the training disorder used to label the MBL phase. The predictions for $L = 14$ and $L = 24$ were obtained by transfer learning.

Discussion — This setup allows to improve on many limitations of the previously considered architectures, namely reducing the training dataset as well as the training region dependencies. Nevertheless, we found that training a DANN is very sensitive to hyperparameters choices (regularization parameter μ, etc..) and chosen NN structure (depth, etc..), hence requiring very good calibration otherwise instabilities can rapidly occur. We also noticed greater variance of the predictions from one instance to another (see appendix B of [Théveniaut & Alet 2019]). Regarding the weight structure in this setup, the PCA in Fig. 3.7 shows that comparatively to single-size and multiple-size, the weights are very different but this does not strictly mean that the learned features are independent of L.

As a last remark, one could easily generalize the use of adversarial components to fight against all at once the dependence on the training region, the discrepancy between samples from the transition region and extremal region of the phase diagram as done in [Huembeli *et al.* 2018] and so on, at the cost of further fine-tuning of extra parameters.

3.4.9 Discussion

Interestingly, our results show consistent behaviours upon changing the input dimension (the classification rule is stable with N_c, see Fig. 3.20) and the system sizes (the classification rule scales appropriately to adapt to the scaling of the first largest p_i, see Fig. 3.21).

The initial goal of this work was to attempt a finite-size study of model Eq. 3.7 using neural networks. Our analysis revealed numerous difficulties: the scaling procedure appeared very sensitive to the neural network hyperparameters (the specific choice of activation function, the addition of dropout or weight decay), as well as the imposed structure (whether an adversarial component is added or not). In addition to that, there is no inherent criterion that allows us to discriminate between these different external choices, and as a matter of fact, we can consider our analysis as a kind of *model exploration* (different machines with the same accuracy have different ways of solving the same task) rather than *model selection* (selecting the machine that achieves the highest accuracy on a given task).

The limitations also arose from the dependence on the particular choice of training dataset, we highlighted that the NN predictions and ultimately the finite-size scaling actually depend on the region of the phase diagram used for training. Moreover when the training dataset includes data from several system sizes, the NN tend to extract average features that do not permit

accurate transfer learning. Including a constraint to fight against this behavior (here in the form of L-invariant adversarial component) improves the situation to a certain extent at the cost of having to fine-tune extra hyperparameters and thus potentially adding more bias in the final estimates.

These limitations occurred even though we provided the best possible input data (i) giving directly the wavefunctions with a controlled compression step and (ii) also in terms of available system size (up to $L = 24$, state-of-the-art in the MBL context). Nevertheless we find that all setups allow to grasp consistent finite-size trends with a sharp behaviour near the critical region, based on a limited amount of disorder realizations. This points towards one of the NN advantages, that is its reduced computational cost compared to conventional methods. Another interesting point (discussed in Sec. 3.4.4 and in the appendix of B of [Théveniaut & Alet 2019]) which we discovered in investigating the contributions to the variance of the prediction is that the network output correlates quite well with the entanglement entropy.

The finite-size scaling led to critical values of h_c and ν quite different from the conventional estimates: $h_c \simeq 3.2$ and $\nu \simeq 0.22$ for the single-size setup, and $h_c \simeq 5-6$ and $\nu \simeq 1.2-1.5$ for the multi-size setups. The finite-size scaling of the MBL transition in model (3.7) (with random disorder) has been shown to be particularly difficult, with system sizes available from exact diagonalization argued to be too small to probe the correct criticality [Khemani *et al.* 2017, Panda *et al.* 2020]. We do not find that the machine learning analysis improves this situation, at least within the setup and input data that we chose. In particular there is no obvious reason to trust more the neural networks final results than the ones reached within the conventional approach. The generic trend that seems to emerge is towards a larger extent of the ETH phase, even though we emphasize that no critical field $h_c(L)$ (obtained for each individual system size L) exceeds the value $h_c \simeq 3.7$ reached from the conventional approach within error bars.

Our thorough finite-size study of this phase transition leads to the conclusion that one always has to be aware of the multiple bias that can possibly arise when using neural networks and its power might be limited to qualitative predictions rather than precise estimations, here for instance finite-size scaling. This is particularly relevant for phase transitions whose nature or universality class is unknown or debated and/or for which the input data has some limitations (*e.g.* in terms of the range of size accessible).

3.5 Application to the ETH-MBL transition in a 2D model

One of the biggest unsettled question about MBL is its fate in systems of physical dimension larger than 1. This question received so far a rather limited amount of attention both from the theoretic and numerical perspectives, given the lack of full understanding of ETH-to-MBL transitions in dimension $d = 1$. The current status of MBL in $d = 2$ can be summarized as follows: there are analytical arguments suggesting that any state eventually reaches thermal equilibrium [De Roeck & Huveneers 2017, De Roeck & Imbrie 2017, Potirniche *et al.* 2019] and numerical and in-lab experiments (limited to finite-time and finite-size) that showed evidence of localization [Thomson & Schiró 2018, Wahl *et al.* 2019, De Tomasi *et al.* 2019], although a recent numerical work [Doggen *et al.* 2020] does not support this.

In the following, I will mostly focus on the results of Sec.IV of [Théveniaut *et al.* 2020] that deals with a machine learning analysis of a possible ETH-MBL transition in a disordered quantum dimer model on the square lattice. In Sec. 3.5.1, the model is introduced. In Sec. 3.5.2,

I will quickly discuss the spectral properties of the model. Finally, Sec. 3.5.3 presents the phase diagram obtained with the same machine learning approach as in last section, though simplified.

3.5.1 The quantum dimer model with random disorder

Our work in [Théveniaut *et al.* 2020] focuses on a two-dimensional quantum dimer model with random potential on the square lattice, which reads:

$$H = -t \sum_{\square} \left(|\square\rangle\langle\square| + |\square\rangle\langle\square| \right) + \sum_{\square_p} V_p \left(|\square\rangle\langle\square| + |\square\rangle\langle\square| \right).$$

The sums run over all plaquettes $\square_p$ of the square lattice, V_p is a random potential different for each plaquette which is drawn uniformly from a box distribution $V_p \in [-V, V]$. We set the kinetic energy scale to $t = 1$. We use shift-invert methods to obtain exact eigenstates in the middle of the spectrum for the system sizes up to $N = 52$ sites. In the following, our ML analysis will be limited to sample with at most 48 sites since not enough samples were available for $N = 52$.

The number of sites N is almost doubled compared to 1d spin chains ($N = 24$ for 1/2-spins) and is a consequence of the constrained nature of model (3.15) since the Hilbert space size scales here as $\sim 1.34^N$ instead of the faster 2^N for spin-1/2 models. Here, the degrees of freedom are dimers that obey the constraint that each site must belong to one and only dimer. This has strong implications on the dynamics since a dimer configuration cannot be obtained by moving one dimer alone. On the one hand, we expect that model (3.15) exhibits slow dynamics due to the constraints, therefore favoring a localized phase. On the other hand, the presence of the local constraints may suggest that one can think of this model as already being in the strongly interacting limit (even without interactions encoded in the Hamiltonian), which would rather favors delocalization and thermalization. The interplay between these two effects was our original motivation to study this model, which was achieved by computing numerous static and dynamical observables (see [Théveniaut *et al.* 2020]).

3.5.2 Results: Spectral statistics

One of the simplest property of a many-body system is its spectral statistics. As discussed in Sec.3.3, the distribution of energy levels can already give us information about the physical system: level repulsion is a footprint of ergodic systems, whereas MBL systems display no correlation between energy levels which translate into Poissonian distribution of energy spacings. The gap ratio (see definition below) was first introduced in [Oganesyan & Huse 2007] in the context of MBL in 1d systems and is known to take well-defined values for Poissonian distributions ($\langle r \rangle \approx 0.38629$) and distributions emanating from the Gaussian Orthogonal random matrix Ensemble (GOE) ($\langle r \rangle \approx 0.5307$).

We define gaps in the many-body spectrum as $s_n = E_n - E_{n-1}$ and consider the consecutive reduced gap ratio $r_n = \frac{\min(s_n, s_{n+1})}{\max(s_n, s_{n+1})}$, for which $0 \leq r \leq 1$. Shown in Fig. 3.26, the average value of $\langle r \rangle = \int_0^1 r P(r) dr$, averaged over eigenstates and disorder realizations, displays an interesting crossover between the two limiting cases, with different system sizes showing an apparent crossing point (see inset of top panel of Fig. 3.26 for square samples $N = 32, 36, 40$) around $V \simeq 15 - 20$. The critical value of the gap ratio $\langle r \rangle^* \simeq 0.392$ is smaller than for the 1d standard MBL

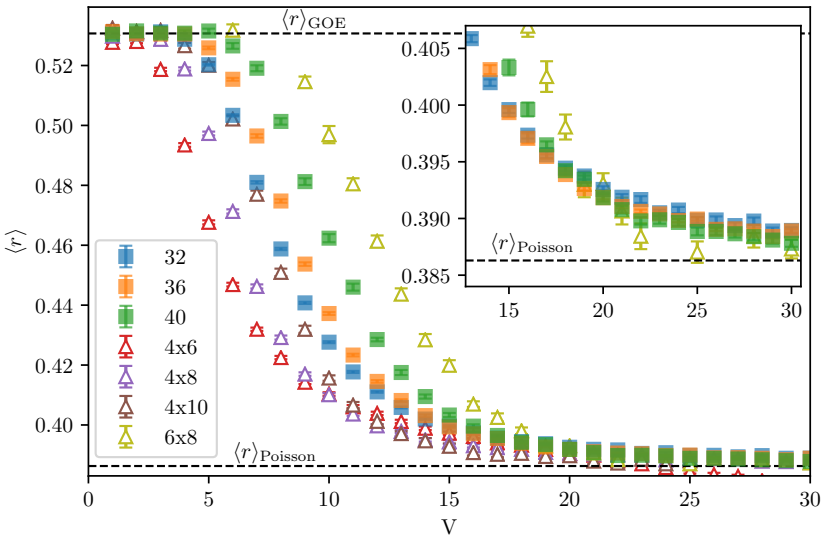


Figure 3.26: Average gap ratio $\langle r \rangle$ for different 2d samples as a function of disorder, for eigenstates located in the middle of the spectrum $\varepsilon = 0.5$.

model [Luitz *et al.* 2015], indicating that the putative transition point looks even closer to the Poissonian localized limit.

In [Théveniaut *et al.* 2020], we also analyzed eigenstate statistics by computing the Kullback-Leibler divergence, dimer occupations and participation entropies as well as entanglement entropies. In all cases, the observables show clear signatures of the two regimes, MBL for large disorder and ETH for small disorder, in the range of system sizes considered. Dynamical probes of localization effects were also studied in [Théveniaut *et al.* 2020] leading to the same conclusion. We now complete this conventional analysis with a ML treatment presented below.

3.5.3 Results: Machine learning this transition

As a complementary approach, we use machine learning techniques to study the quantum dimer model Eq. 3.15 and follow a supervised approach similar to [Schindler *et al.* 2017]. As input data representative of the two phases, we provide entanglement spectra (*i.e.* the eigenvalues λ_i of the reduced density matrix ρ_A) obtained deep in the ETH and the MBL phases and we train a neural network to classify them accordingly (see Fig. 3.27). We previously argued in Sec.3.4.1 that such input data are to be avoided since entanglement spectra are preprocessed quantities that contain a lot of physical knowledge. However, our analysis in Sec. 3.4 concerned MBL in one dimension where the nature of the ETH-to-MBL transition was in question rather than the very existence of the MBL phase which is still debated in 2d. Therefore, our goal is not to determine precisely the location of the critical point (if there is one) but rather to assert the existence of a phase transition. Moreover since it is the first application of ML to this model, we think "helping" a bit the classification task by preprocessing the data (thereby inserting some physical knowledge) is not an issue, and a more refined agnostic study could be left for future investigations.

For this approach, we consider the $N = 32, 36, 40$ square samples and the largest rectangular

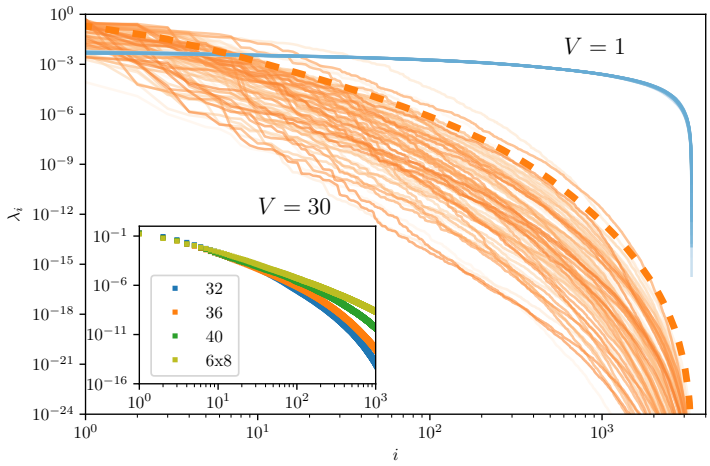


Figure 3.27: Entanglement spectra (eigenvalues λ_i of the reduced density matrix ordered by amplitude) used in the neural network approach to label the ETH ($V = 1$, blue colors) and MBL phase ($V = 30$, orange colors), for the $N = 36$ square sample. For both cases, ~ 100 spectra obtained from different realizations of disorder are represented. The dashed line represents the average of all spectra for $V = 30$. Inset: Higher part of the average entanglement spectra for different sample sizes for $V = 30$. The log-log scale highlights a power-law behavior for the larger eigenvalues λ_i.

sample $N = 6 \times 8$. For each of them, we form a dataset of 10.000 entanglement spectra (2000 for $N = 40, 6 \times 8$) including between 100 and 200 disorder realizations per disorder strength at $V = 1$ for the ETH-labelled phase and $V = 30$ for the MBL-labelled phase. The spectra being rather large (1972 values for $N = 32$, up to 21286 for $N = 6 \times 8$), we found that sorting them allowed for both perfect training and test accuracies for all system sizes. Fig. 3.27 shows the entanglement spectra used in these two limits, the very similar form for various disorder samples in the ETH phase clearly contrasts with the stronger dispersion observed in the MBL phase. In the latter, superposing the entanglement spectra for various samples sizes (inset of Fig. 3.27) highlights that the larger values of the spectrum decay as a power-law, similar to what is found in 1D MBL [Serbyn *et al.* 2016].

We used a fully-connected neural network consisting of one hidden layer of 32 neurons followed by two softmax output neurons. We follow a cross-validation procedure where we randomly selected half of the dataset to form the training dataset, the rest being assigned to the test set. This process is repeated multiple times, generating new training and test partitions each time. This allowed us to track whether the neural networks were overfitting depending on the training conditions. Namely, we checked that data from $V = 1$ and $V = 30$ give perfect training and test accuracies for each system size. We adopted a single-size setup as introduced in Sec.3.4.6 where a neural network is trained and evaluated on data originating from only one system size.

Fig. 3.28 displays features that are consistent with the previous analysis of the gap ratio in the previous section. The left panel displays the average output of the neural network for the different samples as a function of disorder strength. At low V, the machine learning analysis validates a fully-ETH phase (i.e. where all samples are classified ETH) that extends up to a value $V = V_1$, and at large V a fully-MBL phase (with more than 99% accuracy) at large for

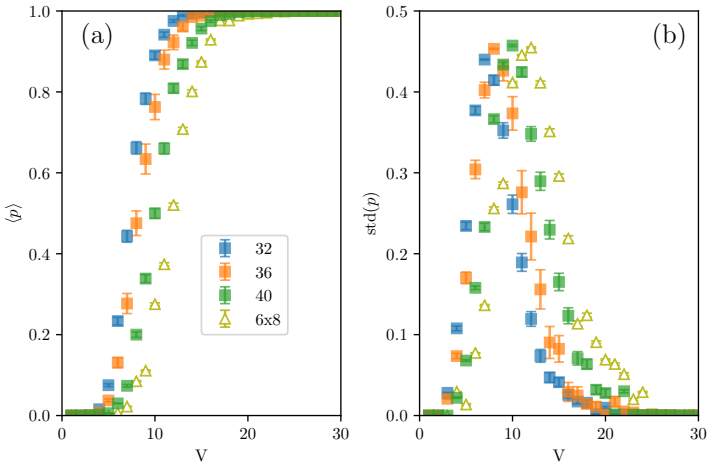


Figure 3.28: As a function of disorder strength V and for different system sizes: (a) mean and (b) standard deviation (right panel) of the neural network output p, defined such that the labels are $p = 0$ ($p = 1$) in the ETH (MBL) samples at $V = 1$ ($V = 30$). The neural network predictions involved 5000 entanglement spectra per disorder (including 100 disorder realizations per disorder). Error bars show standard deviation over 10 instances of neural networks from the cross-validation procedure, stopped after 100 epochs. Square samples are color highlighted.

$V \geq V_2$ ($V_1 \simeq 6$ and $V_2 \simeq 20$ for the largest $N = 6 \times 8$ sample). Notice how these bounding values (in particular V_1) shift to higher values of disorder with system size. This reflects that ETH is easier to disrupt on a too small sample, in perfect agreement with the trend in all other observables discussed in [Théveniaut *et al.* 2020]. Similar to what was observed in the 1d MBL transition, we find no crossing point with system size.

The right panel of Fig. 3.28 shows the standard deviation of the neural network output as a function of disorder. The standard deviation is low in both limits where the phases are well distinguished (at low and large V), and peaks at an intermediate value of disorder. The location of the peak (which shifts with system size) is the point where the neural network has most difficulties to classify phases. It can be interpreted as a finite-size estimate of a possible transition point. Notice the similarity between the standard deviation of the neural network output and the standard deviation of the entanglement entropy (Fig. 8 in [Théveniaut *et al.* 2020]), in particular the positions of the peaks are almost the same for both quantities for the different sample sizes.

In conclusion of this section, we find that a neural network only fed with entanglement spectra is able to learn how to correctly distinguish the ETH and MBL regimes for the quantum dimer model with random potential Eq. 3.15 as well as to provide finite-size estimates of the transition point between the two. This automated method gives results in good qualitative agreement with the analysis based on more standard, feature-engineered, estimators of the phases presented in [Théveniaut *et al.* 2020]. One noticeable interest of the neural network analysis (already pinpointed earlier [Schindler *et al.* 2017, Venderley *et al.* 2018]) is that the required amount of data and overall computational cost is considerably lower than with more traditional observables to obtain approximately similar quality of prediction: for instance good statistics on the gap ratio (Fig. 3.26) requires $\simeq$ 100 times more realizations of disorder than with the machine learning analysis.

3.5.4 Discussion

All data presented in [Théveniaut *et al.* 2020], both for eigenstate or dynamical properties, can be interpreted as consistent with the existence of two distinct phases: an ETH phase at low disorder and a many-body localized phase at strong disorder, which are separated by a transition located around disorder strength $V \simeq 15 - 20$. The ML analysis of entanglement spectra presented in Sec. 3.5.3 provides the same conclusion.

Since MBL in 2d is a controversial topic, let us now do a critical analysis of these results. Our evidence for a MBL transition in the 2d quantum dimer model comes from numerical simulations on finite lattices. Of course, finite-size simulations can always be argued to artificially detect a MBL phase even when only a ETH phase occurs in the thermodynamic limit. We would like to emphasize that the level of numerical evidence for a MBL transition in the 2d quantum dimer model is similar to the one obtained for the standard model of MBL in one dimension [Pal & Huse 2010, Luitz *et al.* 2015], with similar or larger Hilbert space sizes and time scales probed. This is of course not a definite proof that a MBL transition occurs in the thermodynamic limit. What is perhaps more important with respect to potential experiments is the fact that even if the behavior at large values of disorder (say $V \geq 25$) is ultimately ergodic, the time scales and / or system sizes needed to probe ergodicity would be extremely large. An experimental realization of Eq. 3.15 or of a similar constrained model in 2d would see localization for all practical purposes on the time scales available in the lab. For one-dimensional MBL, the larger linear length scales that can be reached have been argued (see e.g. [Khemani *et al.* 2017]) not to be large enough to provide correct estimates of asymptotic critical behavior. The situation is likely to be the same here. With these numerical limitations in mind, we can nevertheless observe that critical values of the gap ratio of eigenstates are closer to their Poisson than their ETH limits, indicating that the putative transition point is even less ergodic than for the one-dimensional MBL transition in the random-field Heisenberg spin chain [Luitz *et al.* 2015].

There are several perspectives opened up by our work. First, the roadmap to 2d MBL can be exploited using the QDM on other lattices, allowing to test for universality and to search for other features. A recent investigation [Pietracaprina & Alet 2020] considered the honeycomb lattice, which has an effective smaller local Hilbert space allowing to reach larger samples, and obtained results similar to ours. The QDM on the kagome lattice is also an interesting case as it possesses conserved Z_2 quantum numbers, allowing the exciting possibility of 2d topological order in MBL states [Moessner & Sondhi 2001]. The possibility of MBL in other quantum constrained models such as quantum ice or loop models also provide an interesting follow-up.

Regarding the ML part, it would be intriguing to see if the machine learning techniques used in Sec. 3.5.3 would be able to distinguish 1d from 2d MBL, and if not, to use neural networks trained on 1d spin chain models to probe 2d MBL (transfer learning). Nevertheless, we think it is probable that the same limitations brought to light for the 1d case in Sec. 3.4 will eventually appear upon closer inspection of the predictions. Namely, there is no reason to expect that the dependence of the ML results with the dataset or with the neural-network architecture will be lower in model 3.15.

Chapter 4

Neural-network quantum states

Contents

This chapter deals with the variational approximation of quantum many-body ground states. While in previous chapter the problem of mapping phase diagram was recast as a conventional supervised learning task, here instead machine learning was used as a source of inspiration by Carleo and Troyer in [Carleo & Troyer 2017] when they examined the ability of neural networks to encode quantum wave functions.

In Sec. 4.1, the set of neural-network quantum states (NQS) and the particular family we use, restricted Boltzmann machines, are defined and we will highlight some of their physical properties. Sec. 4.2 will introduce the method of variational Monte Carlo (VMC) and Sec. 4.3 will focus on a projection technique called reptation Quantum Monte Carlo (RQMC) that are important numerical tools in condensed matter physics. Sec.4.4 will present the results we obtained on a two-dimensional constrained model of hardcore bosons using NQS applied in the framework of both variational and projection methods.

4.1 Neural-network quantum states

In the era of the development of quantum computers, one may think that it is a bit anachronistic to try to improve on classical algorithms to study quantum materials. However it is still not clear

whether there exists a quantum advantage for approximate optimization problems[Preskill 2018], which means that approximating quantum ground-states with classical algorithms is still of great relevance. At the same time, the industrial and academic interest in the field of machine learning has exploded and brought about a lot of algorithmic improvements, efficient implementations and hardware[Jouppi *et al.* 2017]. As a result, one can consider that machine learning arguably contains the most powerful set of numerical techniques for the approximation of high-dimensional functions nowadays. This certainly explains the surge of interest of ML in condensed matter physics and in particular the motivation of Carleo and Troyer to design neural-network based quantum wave functions[Carleo & Troyer 2017].

Let us denote $s_1, \dots, s_N$ the quantum numbers of a system of N particles, one can write any state in the basis set of elements $|s_1, \dots, s_N\rangle$, which leads to the following definition of neural-network quantum states (NQS):

$$|\Psi_{\text{NQS}}\rangle = \sum_{s_1,\cdots,s_N} f(s_1, \dots, s_N)|s_1, \dots, s_N\rangle \tag{4.1}$$

where f is a neural network, i.e. a high-dimensional function that takes an input of dimension N and returns a complex number (note that in conventional ML tasks, neural networks usually manipulate real numbers). Eq. (4.1) defines a large ensemble of states that is parametrized by the choice of neural-networks (FFNN, CNN, etc..), their specific architecture (depth, etc..) and the values of their internal weights.

As we will see in Sec. 4.2, these states can serve as variational ansatz to approximate ground-states of quantum many-body systems. Another possible application is quantum state tomography where one wants to reconstruct the many-body wave function of a system realized in an experiment having only access to projective measurements in a given basis[Carrasquilla *et al.* 2019, Torlai *et al.* 2018]. One can notice that the state in Eq. 4.1 is defined up to a particular basis set, the natural basis arises from the experimental constraints in tomography, but there is more freedom in VMC as will see later.

More importantly, the universal approximation theorems of machine learning ensure that any physical state can be approximated with arbitrary precision with a NQS. However this does not exclude the case where the size of the neural-network has to scale exponentially with system size, therefore one of the primary focus of the field was to investigate the *efficiency* of NQS, in other words the possibility to encode quantum states with neural networks growing only polynomially with system size.

4.1.1 Restricted Boltzmann Machine

The first NQS considered in [Carleo & Troyer 2017] was based on a restricted Boltzmann machine (RBM). As defined in Sec. 2.3.3 of chapter 2, RBMs are energy-based models that contain visible and hidden neurons coupled to each other as shown in Fig. 4.1.

Since the neuron values are usually binary, it is particularly natural to use this parametrization for spin-1/2 systems, with local magnetizations $\sigma_i^z = \pm 1$ chosen as quantum numbers. More precisely, for a spin-1/2 system of N constituents, any quantum state $|\Psi\rangle$ can be written in the basis set of states $|\sigma_1^z\rangle \otimes \cdots \otimes |\sigma_N^z\rangle$ (with $\hat{\sigma}_i^z|\sigma_i^z\rangle = \sigma_i^z|\sigma_i^z\rangle$). The RBM ansatz consists in parametrizing the complex amplitude on one this basis set, i.e. $\langle\sigma_1^z \cdots \sigma_N^z|\Psi\rangle$, with the partition

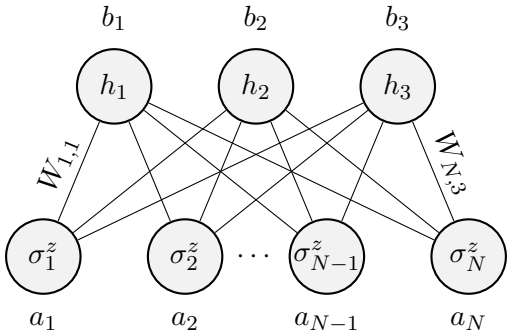


Figure 4.1: A RBM having N visible units taking the values of the local magnetizations $\sigma_i^z = \pm 1$ of a spin-1/2 system and 3 hidden nodes $h_j = \pm 1$. Some of its internal parameters are shown: visible biases a_i, hidden biases b_j and connection weights $W_{i,j}$.

function of a RBM as follows:

$$\Psi_{\mathrm{RBM}}(\sigma_1^z, \dots, \sigma_N^z) = \sum_{(h_1,\dots,h_M)\in\{-1,1\}^M} \exp\left(\sum_{i=1}^{N} a_i\sigma_i^z + \sum_{j=1}^{M} b_j h_j + \sum_{i=1}^{N}\sum_{j=1}^{M} W_{ij}\sigma_i^z h_j\right) \tag{4.2}$$

where $\mathbf{a} = (a_i)_{i=1\dots N} \in \mathbb{C}^N$, $\mathbf{b} = (b_j)_{j=1\dots M} \in \mathbb{C}^M$ and $\mathbf{W} = (W_{ij})_{i=1\dots N, j=1\dots M} \in \mathbb{C}^{N\times M}$ are the complex variational parameters. Note that if these parameters were real, the wave function would be real and positive and therefore could not encode a fermionic state with a sign structure. Due to the absence of intralayer connections (visible-visible, hidden-hidden) in the RBM, the expression above can be factored out:

$$\Psi_{\mathrm{RBM}}(\sigma_1^z, \dots, \sigma_N^z) = \exp\left(\sum_{i=1}^{N} a_i\sigma_i^z\right) \prod_{j=1}^{M} 2\cosh\left(b_j + \sum_{i=1}^{N} W_{ij}\sigma_i^z\right) \tag{4.3}$$

Efficiency of RBM states. The parametrization in Eq. 4.3 involves a total of $NM + M + N$ variational parameters which, provided the number of hidden nodes M is a polynomial function of the system size N, results in an ansatz of polynomial complexity. The expressive power of RBM states (which physical states can they efficiently approximate?) has been extensively studied in the past years, with a particular focus on how they compare to tensor-network states. Indeed, matrix product states (MPS) are considered state-of-the-art for 1D gapped systems[Verstraete *et al.* 2008] where the entanglement is small, but the situation is less favorable in 2D since there exist area-law (for entanglement entropy) states that cannot be efficiently approximated by PEPS[Ge & Eisert 2015] – the 2D generalization of MPS. A first striking result was obtained in [Deng *et al.* 2017b] as the authors showed that a RBM can encode area-law as well as volume-law states, in contrast to tensor-network states that can only sustain low entanglement. Moreover, this could be achieved very efficiently, the number of RBM parameters scaling only linearly with system size. It was also proven that RBMs can exactly represent topological states[Deng *et al.* 2017a, Clark 2018] and that there exist RBMs of specific weight connectivity that map exactly to tensor-network states[Glasser *et al.* 2018, Chen *et al.* 2018]. Despite these encouraging results, RBMs as defined in Eq. 4.2 are shallow networks (they can be mapped to

a two-layer feed-forward neural networks) and thus limited. In particular, there exist a class of many-body ground states that cannot be efficiently approximated by RBMs[Gao & Duan 2017].

Advantages of depth. Contrary to machine learning where the advantage of depth still lacks rigorous arguments, the situation is quite the opposite in quantum many-body physics. The theoretical superiority of deep architectures was studied in [Gao & Duan 2017] where they provide a rigorous proof that deep neural networks can efficiently represent most physical states, including ground states of many-body Hamiltonians. Moreover, deep NQS based on CNN or RNN can sustain entanglement polynomially more efficiently than shallow RBM states[Levine *et al.* 2019]. Finally, deep Boltzmann machines – a deep generalization of RBM – are provably capable of exactly representing quantum many-body states[Carleo *et al.* 2018]. However, one has to keep in mind that depth comes with the price of a higher computational cost which can make these deep NQS impractical in practice.

4.1.2 Implementing symmetries

Hamiltonians often have symmetries that their eigenstates inherit. Consequently, variational quantum states that directly implement the symmetries of the model are expected to perform best. For spatial symmetries in lattice models, a NQS based on CNN for example can easily enforce translation-invariance[Choo *et al.* 2019]. Other symmetries like fermionic symmetry are however more difficult to incorporate in the parametrization.

We follow the construction employed in [Carleo & Troyer 2017, Fabiani & Mentink 2019] to design a symmetric RBM wave function. Let us consider a symmetry group containing S linear transformations T_s that commute with the physical Hamiltonian, for instance translations, reflections or rotations of a lattice. We denote the transformed spin configurations $T_s(\sigma_i^z) = \widetilde{\sigma_i^z}(s)$. We would like to have a RBM parametrization such that $\Psi(\sigma_1^z, \dots, \sigma_N^z) = \Psi(\widetilde{\sigma_1^z}(s), \dots, \widetilde{\sigma_N^z}(s))$ (up to a phase factor) for every transformation T_s and all configurations $\boldsymbol{\sigma}^z = (\sigma_1^z, \dots, \sigma_N^z)$. This can be achieved by considering a larger RBM having $S \times N$ visible units fed with the values of $\boldsymbol{\sigma}^z$ and all its transformed copies $T_1\boldsymbol{\sigma}^z, \dots, T_S\boldsymbol{\sigma}^z$, coupled to $S \times M$ hidden units but with sparse weight connectivity and parameter sharing across the network as shown in Fig. 4.2.

This corresponds to the symmetric parametrization introduced in Eq. (S13) of the supplementary material of [Carleo & Troyer 2017] (albeit with minor differences[1]):

$$\Psi^{\text{sym}}_{\text{RBM}}(\sigma_1^z, \dots, \sigma_N^z) = \sum_{(h_{f,s}) \in \{-1,1\}^{M \times S}} \prod_{s=1}^{S} \exp\left(\sum_{i=1}^{N} a_i \widetilde{\sigma_i^z}(s) + \sum_{f=1}^{M} b^{(f)} h_{f,s} + \sum_{f=1}^{M} \sum_{i=1}^{N} \widetilde{\sigma_i^z}(s) W_i^{(f)} h_{f,s} \right) \tag{4.4}$$

[1] We corrected some typos in Eq.(S13), in particular their definition of the visible bias $a^{(f)}$ is such that $\sum_{f,s,i} a^{(f)} \widetilde{\sigma_i^z}(s) = \left(\sum_f a^{(f)}\right) \sum_{s,i} \widetilde{\sigma_i^z}(s) \equiv A \sum_{s,i} \widetilde{\sigma_i^z}(s)$. We changed the definition of the visible bias to the more natural form: $\sum_{s,i} a_i \widetilde{\sigma_i^z}(s)$.

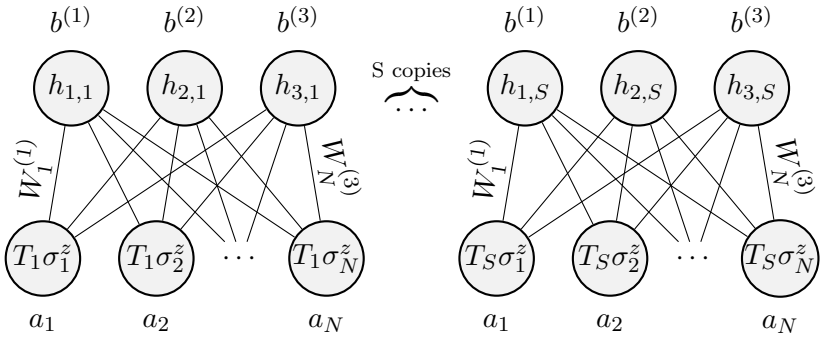


Figure 4.2: A restricted Boltzmann machine that is invariant under the action of the transformations T_s corresponding to S symmetries of the system. This is done by considering S copies of the same RBM whose parameters are shared across the network, this gives rise to $S \times N$ visible units corresponding to the S transformed copies of the "spin" configurations $T_s\boldsymbol{\sigma}^z$ and $S \times M$ hidden units (here $M = 3$). Parameter sharing manifests in the fact that the same biases a_i, $b^{(f)}$ and weights $W_i^{(f)}$ are applied to each spin configuration $T_s\boldsymbol{\sigma}^z$.

As done previously, it is possible to trace out the hidden variables, which gives:

$$\Psi_{\mathrm{RBM}}^{\mathrm{sym}}(\sigma_1^z, \dots, \sigma_N^z) = \prod_{s=1}^{S} \exp\left(\sum_{i=1}^{N} a_i \widetilde{\sigma_i^z}(s)\right) \prod_{f=1}^{M} 2\cosh\left(b^{(f)} + \sum_{i=1}^{N} W_i^{(f)} \widetilde{\sigma_i^z}(s)\right) \tag{4.5}$$

A symmetric RBM has $NM + M + N$ variational parameters which is the same number as for a non-symmetric RBM. However, the number of hidden nodes in non-symmetric RBMs is usually chosen to scale with N as $M \equiv \alpha N$, whereas it is chosen constant for symmetric RBM $M \equiv \alpha$. This means that *symmetric RBM ansatz will have in practice less variational parameters than non-symmetric RBM ansatz*. This last point will be discussed in Sec. 4.4.

4.2 Variational Monte Carlo

The idea of variational approximation was introduced in the early days of quantum mechanics when Heitler and London proposed a simplified expression of the ground state of the H_2 molecule. Roughly speaking, variational approximation boils down to approximating a target state with a guess wave function (*ansatz*) which depends on a preferably small number of variational parameters, this way studying the target state is made analytically and/or numerically tractable. Many wave functions of electronic systems have been proposed: ones that assume that electrons are independent i.e. using the so-called Hartree-Fock approximation[Slater 1930], ones that include the effect of on-site electron repulsion also known as Gutzwiller wave functions[Gutzwiller 1963], which can be further modified to account for density-density correlations in the form of Jastrow factors[Jastrow 1955]. In Sec. 4.2.3, we will see how NQS can be used as variational wave functions. This section follows to a great extent chapters 5 and 6 of [Becca & Sorella 2017].

4.2.1 Variational principle

Let us say one wants to find a good approximation of the ground-state $|E_0\rangle$ of a given Hamiltonian H using a variational state $|\Psi_{\boldsymbol{\theta}}\rangle$. How can one find the parameters of the wave-function $\boldsymbol{\theta}$ such that $|\Psi_{\boldsymbol{\theta}}\rangle$ is close to $|E_0\rangle$? The first part of the solution is provided by the variational principle derived below (we use $|\Psi_{\boldsymbol{\theta}}\rangle = \sum_i a_i |E_i\rangle$ in the eigenbasis of H):

$$\varepsilon \equiv E_{\boldsymbol{\theta}} - E_0 = \frac{\langle\Psi_{\boldsymbol{\theta}}|H|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} - E_0 \tag{4.6}$$

$$= \sum_i |a_i|^2 E_i - E_0 \tag{4.7}$$

$$= \sum_i |a_i|^2 (E_i - E_0) \tag{4.8}$$

$$\geq 0 \tag{4.9}$$

where we used the normalization condition $\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle = \sum_i |a_i|^2 = 1$ and the fact that $E_0 \leq E_i$. In particular, this shows that *the energy of any variational state* $|\Psi_{\boldsymbol{\theta}}\rangle$ *provides an* upper bound *on the true ground state energy* E_0. Therefore, by minimizing $E_{\boldsymbol{\theta}}$, one can expect that the variational wave function will become a better approximation of the true ground-state wave function $|E_0\rangle$. This statement can be made more precise by considering the energy gap $\Delta \equiv E_1 - E_0$ with respect to the first excited state. Indeed, one can obtain a better lower bound in Eq. 4.8 since $E_i - E_0 \geq \Delta$ for $i \neq 0$, hence:

$$\varepsilon \geq \Delta \sum_{i\neq 0} |a_i|^2 \tag{4.10}$$

$$\frac{\varepsilon}{\Delta} \geq 1 - |a_0|^2 = 1 - \langle E_0|\Psi_{\boldsymbol{\theta}}\rangle \tag{4.11}$$

One sees that if the energy error ε is small compared to the gap, the overlap $\langle E_0|\Psi_{\boldsymbol{\theta}}\rangle$ is guaranteed to be close to 1.

If the parametrization $|\Psi_{\boldsymbol{\theta}}\rangle$ is too simple, the true target ground-state could potentially be out of reach and in general this makes this method *biased*. In particular, difficult cases arise when many incompatible assumptions are possible, for instance when there exist states with different physical properties but close in energy as in the frustrated $J_1 - J_2$ Heisenberg model[Figueirido *et al.* 1990, Kotov *et al.* 1999, Zhitomirsky & Ueda 1996]. Nevertheless an important advantage of the variational approach is that it does not suffer from the sign problem contrary to quantum Monte Carlo methods.

Calculating energy and diagonal observables. Let us denote a complete orthogonal and normalized basis set $\{|x\rangle\}$ of the many-body Hilbert space. The energy expectation value $E_{\boldsymbol{\theta}}$ can be computed as follows (using a resolution of the identity $\sum_x |x\rangle\langle x| = I$):

$$E_{\boldsymbol{\theta}} = \frac{\langle\Psi_{\boldsymbol{\theta}}|H|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \sum_x \frac{|\Psi_{\boldsymbol{\theta}}(x)|^2}{\sum_x |\Psi_{\boldsymbol{\theta}}(x)|^2} e_L(x) \tag{4.12}$$

where we introduced the local energy $e_L(x)$:

$$e_L(x) = \frac{\langle x|H|\psi_{\boldsymbol{\theta}}\rangle}{\langle x|\psi_{\boldsymbol{\theta}}\rangle} = \sum_{x'} H_{x,x'} \frac{\Psi_{\boldsymbol{\theta}}(x')}{\Psi_{\boldsymbol{\theta}}(x)} \tag{4.13}$$

Note that the Hamiltonian H is in general a local operator which means that the sum $\sum_{x'}$ is tractable since there are at most $O(N)$ non-zero elements $H_{x,x'}$ for a given x, with N being the number of lattice sites. For an observable $\mathcal{O}$ diagonal in the basis $\{|x\rangle\}$ (i.e. $\langle x|\mathcal{O}|x'\rangle = \delta_{x,x'}\mathcal{O}_x$), one can similarly compute the expectation of $\mathcal{O}$ on $|\Psi_{\boldsymbol{\theta}}\rangle$ as follows:

$$\langle\mathcal{O}\rangle = \frac{\langle\Psi_{\boldsymbol{\theta}}|\mathcal{O}|\Psi_{\boldsymbol{\theta}}\rangle}{\langle\Psi_{\boldsymbol{\theta}}|\Psi_{\boldsymbol{\theta}}\rangle} = \sum_x \frac{|\Psi_{\boldsymbol{\theta}}(x)|^2}{\sum_x |\Psi_{\boldsymbol{\theta}}(x)|^2}\mathcal{O}_x \tag{4.14}$$

Given that $\mathcal{P}(x) \equiv \frac{|\Psi_{\boldsymbol{\theta}}(x)|^2}{\sum_x |\Psi_{\boldsymbol{\theta}}(x)|^2}$ defines a probability distribution, it is clear that energy and diagonal observables can be evaluated statistically with a Markov chain Monte Carlo (MCMC) procedure. By performing a random walk in configuration space (here $\{|x\rangle\}$) with a probability of selecting and accepting the next configuration such that detailed balance is satisfied, a sequence of configurations $\{x_i\}$ can be generated and will be distributed as $\mathcal{P}(x)$. The acceptance probability follows the usual Metropolis scheme:

$$A(x_i \to x_{i+1}) = \min\left(1, \left|\frac{\Psi_{\boldsymbol{\theta}}(x_{i+1})}{\Psi_{\boldsymbol{\theta}}(x_i)}\right|^2\right) \tag{4.15}$$

The selection rule is usually chosen such that the ratio $\Psi(x_{i+1})/\Psi(x_i)$ can be evaluated quickly. Also it must be ergodic meaning that any configuration x can be reached from any starting x_0 in configuration space. All in all, the energy (Eq. 4.12) and diagonal observables (Eq. 4.14) can be estimated from a sequence of configurations $\{x_i\}_{i=1\ldots N_{\text{samples}}}$ as follows:

$$E_{\boldsymbol{\theta}} \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} e_L(x_i) \tag{4.16}$$

$$\langle\mathcal{O}\rangle \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} \mathcal{O}_L(x_i) \tag{4.17}$$

The zero-variance property. An important feature of VMC is the fact that whenever a variational state coincides with an exact eigenstate of the Hamiltonian, the expectation value of the energy variance $\langle(H-E)^2\rangle$ vanishes. This originates from the property of the local energy $e_L(x)$:

$$e_L(x) = \frac{\langle x|H|\Psi_{\boldsymbol{\theta}}\rangle}{\langle x|\Psi_{\boldsymbol{\theta}}\rangle} = E\frac{\langle x|\Psi_{\boldsymbol{\theta}}\rangle}{\langle x|\Psi_{\boldsymbol{\theta}}\rangle} = E \tag{4.18}$$

which becomes constant in that case and thereby shows that the random variable $e_L(x_i)$ does not fluctuate. In some VMC settings, this property is used and minimization of the energy variance is attempted instead of minimization of the energy.

4.2.2 Energy minimization

Gradient descent can be applied to minimize $E_{\boldsymbol{\theta}}$, this means the parameters of the variational state are updated according to:

$$\theta_j \leftarrow \theta_k - \lambda\frac{\partial E_{\boldsymbol{\theta}}}{\partial\theta_k} \tag{4.19}$$

with the learning rate λ being sufficiently small to ensure convergence. Note that one can include higher-order terms in Eq. 4.19 (though it becomes too expensive in the majority of cases) or use algorithms like Adam[Kingma & Ba 2015], Adadelta[Duchi *et al.* 2011] that adapt the learning rate as done in ML (see Sec. 2.3.4 in chapter 2). As shown in [Becca & Sorella 2017], evaluating $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$ requires computing the logarithmic derivative of $\Psi_{\boldsymbol{\theta}}$, which we denote O_k which reads:

$$O_k(x) = \frac{\partial \log \Psi_{\boldsymbol{\theta}}(x)}{\partial \theta_k} = \frac{1}{\Psi_{\boldsymbol{\theta}}(x)} \frac{\partial \Psi_{\boldsymbol{\theta}}(x)}{\partial \theta_k} \tag{4.20}$$

The detailed derivation can be found in [Becca & Sorella 2017] and the final results include the local energy $e_L(x)$ defined in Eq. 4.13:

$$\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k} \approx 2\Re \left[\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} e_L^*(x_i) \left(O_k(x_i) - \overline{O_k} \right) \right] \tag{4.21}$$

$$\overline{O_k} = \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} O_k(x_i) \tag{4.22}$$

Stochastic reconfiguration. The stochastic reconfiguration (SR) algorithm was introduced in [Sorella 1998] and uses information about the geometry of the parameter space to improve the stability of the optimization. SR is also known as natural gradient[Amari 1998] in the ML community and stems from the observation that the gradient descent update rule (Eq. 4.19) assumes that the parameter space is Euclidean, i.e. every dimension corresponding to a parameter θ_k is equivalent. However in most cases many non-equivalent variational parameters enter the ansatz in a highly non-linear fashion. Put in other words, a change of parameter $\theta_k + \delta\theta$ results in a wavefunction $|\Psi_{\boldsymbol{\theta}+\delta\theta \mathbf{e}_k}\rangle$ that can be very different from $|\Psi_{\boldsymbol{\theta}+\delta\theta \mathbf{e}_{k'}}\rangle$, although the change in parameter $\theta_{k'}$ was done with the same magnitude $\delta\theta$. This calls the consideration of the non-Euclidean metric δs that defines a distance with respect to change of the wavefunction, i.e.

$$\delta s^2 = \min_{\phi} \| \exp(-i\phi) \Psi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}} - \Psi_{\boldsymbol{\theta}} \|^2 \tag{4.23}$$

where the presence of ϕ is required as we do not want to distinguish wave functions that differ only by a phase factor. This distance can be related to the original Euclidean metric via the following relation[Sorella 1998]:

$$\delta s^2 = \sum_{k,k'} S_{k,k'} \delta\theta_k \delta\theta_{k'} \tag{4.24}$$

where the covariance S matrix can be obtained stochastically with the relation:

$$S_{k,k'} \approx \Re \left[\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} (O_k(x_i) - \overline{O_k})(O_{k'}(x_i) - \overline{O_{k'}}) \right] \tag{4.25}$$

Finally, the parameter update takes the form[Sorella 1998]:

$$\theta_{k+1} = \theta_k - \lambda \sum_{k'} S^{-1}_{k,k'} \frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_{k'}} \tag{4.26}$$

In practice, the matrix S can be non-invertible and explicit regularization may be needed. One possibility is to add a small diagonal matrix of the form $S^{\text{reg}}_{k,k'} = S_{k,k'} + \lambda_t \delta_{k,k'} S_{k,k}$ as in [Carleo & Troyer 2017] with λ_t decaying with the number of VMC iterations t. In addition to that, it is possible to avoid inverting the whole matrix S and instead exploit the structure of S combined with conjugate-gradient methods to iteratively compute the pseudo-inverse S^{-1}.

Implementation. In summary, each gradient descent step involves:

1. generating a sequence of N_{samples} independent configurations $\{x_i\}$ obtained after equilibrating a MCMC according to the probability distribution $\mathcal{P}(x) = \frac{|\Psi_{\boldsymbol{\theta}}(x)|^2}{\sum_x |\Psi_{\boldsymbol{\theta}}(x)|^2}$,
2. computing the values of the local energy $e_L(x_i)$ (Eq. 4.18), the logarithmic derivatives of $\Psi_{\boldsymbol{\theta}}$, $O_k(x_i)$ (Eq. 4.20) and $\overline{O_k}$ (Eq. 4.22) for all samples x_i and all the variational parameters θ_k,
3. computing $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$ (Eq. 4.21) and $S_{k,k'}$ (4.25) for all the variational parameters θ_k,
4. if stochastic reconfiguration is used, computing the inverse of the S matrix,
5. updating the parameters θ_k according to Eq. 4.26.

4.2.3 RBM as variational wave functions

The seminal work of Carleo and Troyer[Carleo & Troyer 2017] stimulated a number of contributions[Nomura *et al.* 2017, Fabiani & Mentink 2019, Ferrari *et al.* 2019, Inack *et al.* 2018, Pilati & Pieri 2020] that showed that RBM-type wave functions can provide an accurate variational description of ground states of several 1d and 2d bosonic and fermionic quantum systems in the framework of variational Monte Carlo. Other NQS have been considered and produced excellent results: CNN in [Choo *et al.* 2019], RNN in [Hibat-Allah *et al.* 2020] or autoregressive networks in [Sharir *et al.* 2020]. The VMC method essentially relies on the computation of the log-derivatives O_k and the local energy e_L where the ratio $\Psi(x')/\Psi(x)$ appear. When $|\Psi_{\boldsymbol{\theta}}\rangle$ is parametrized by a RBM, the expression of O_k is given in Tab. 4.1.

non-symmetric RBM (Eq. 4.2)	symmetric RBM (Eq. 4.4)
$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial a_i}(\boldsymbol{\sigma}^z) = \sigma_i^z$	$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial a_i}(\boldsymbol{\sigma}^z) = \sum_{s=1}^{S} \widetilde{\sigma_i^z}(s)$
$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial b_j}(\boldsymbol{\sigma}^z) = \tanh\left[\theta_j(\boldsymbol{\sigma}^z)\right]$	$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial b^{(f)}}(\boldsymbol{\sigma}^z) = \sum_{s=1}^{S} \tanh\left[\theta_{f,s}(\boldsymbol{\sigma}^z)\right]$
$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial W_{ij}}(\boldsymbol{\sigma}^z) = \sigma_i^z \tanh\left[\theta_j(\boldsymbol{\sigma}^z)\right]$	$\frac{1}{\Psi(\boldsymbol{\sigma}^z)} \frac{\partial \Psi}{\partial W_i^{(f)}}(\boldsymbol{\sigma}^z) = \sum_{s=1}^{S} \widetilde{\sigma_i^z}(s) \tanh\left[\theta_{f,s}(\boldsymbol{\sigma}^z)\right]$

Table 4.1: Log-derivatives of two types of RBM wave functions with respect to all variational parameters. We defined angles $\theta_j(\boldsymbol{\sigma}^z) = b_j + \sum_i W_{ij}\sigma_i^z$ for non-symmetric RBMs and $\theta_{f,s}(\boldsymbol{\sigma}^z) = b^{(f)} + \sum_i W_i^{(f)} \widetilde{\sigma_i^z}(s)$ for symmetric RBMs.

We refer to Sec. A.1 of appendix A for a detailed discussion of the complexity of the VMC algorithm with RBM wave functions. The dominant cost is shown to be $O(\alpha N^2 \times N_{\text{samples}})$ for both $M \equiv \alpha N$ non-symmetric RBMs and $M \equiv \alpha$ symmetric RBMs.

4.3 Projection methods

Projection methods rely on the *power method* which consists in repeatedly applying the Hamiltonian H on a given state $|\Psi_{\text{init}}\rangle$, after a large enough number of projections, the projected state is expected to be close to the eigenstate of largest absolute eigenvalue of H. In other words, we have:

$$\lim_{p\to\infty} (\lambda I - H)^p |\Psi_{\text{init}}\rangle \propto |E_0\rangle \tag{4.27}$$

where $|E_0\rangle$ is the ground state wave function and the value of λ (which has no consequences on the physics) is such that the largest absolute eigenvalue of $(\lambda I - H)$ is the ground-state energy E_0. The successive projections have the effect of filtering out the highest-energy states because of the following relations (using $|\Psi_{\text{init}}\rangle = \sum_i a_i |E_i\rangle$):

$$(\lambda I - H)^p |\Psi_{\text{init}}\rangle = \sum_i a_i (\lambda - E_i)^p |E_i\rangle \tag{4.28}$$

$$= a_0(\lambda - E_0)^p \left(|E_0\rangle + \sum_{i\neq 0} \frac{a_i}{a_0} \left(\frac{\lambda - E_i}{\lambda - E_0} \right)^p |E_i\rangle \right) \tag{4.29}$$

in particular with a suitable choice of λ such that $|\lambda - E_i| < |\lambda - E_0|$, the method converges exponentially fast to the ground-state wave function (the factor $a_0(\lambda - E_0)^p$ being irrelevant) as long as $|\Psi_{\text{init}}\rangle$ is not orthogonal to the ground state ($a_0 \neq 0$). Note that in accordance with the variational principle, the energy of the projected state is also an upper bound of the true ground state energy. Contrary to VMC, projection methods are exact by nature for p large enough which makes them particularly appealing for the study of strongly-correlated models. One way to use the power method for large systems is to consider the projection operations stochastically as it is done in the reptation Quantum Monte Carlo method presented below.

4.3.1 Reptation QMC

An interesting approach called *reptation Quantum Monte Carlo* (RQMC) was introduced in [Baroni & Moroni 1999] that uses a path-integral representation leading to a method conceptually more simple than the previously used many walkers formulation[Calandra-Buonaura & Sorella 1998]. The aim of RQMC is to define a Markov process that allows to sample the following partition function:

$$Z = \langle \Psi_{\text{init}} | (\lambda I - H)^p | \Psi_{\text{init}} \rangle \tag{4.30}$$

where $|\Psi_{\text{init}}\rangle$ is a given state that can be random or the best variational state obtained from VMC calculations. The impact of the choice of $|\Psi_{\text{init}}\rangle$ will be evaluated later in this section. By inserting $p+1$ resolutions of unity in Eq. 4.30, we obtain:

$$Z = \sum_{x_0,\ldots,x_p} \langle \Psi_{\text{init}} | x_0 \rangle \langle x_0 | (\lambda I - H) | x_1 \rangle \cdots \langle x_{p-1} | (\lambda I - H) | x_p \rangle \langle x_p | \Psi_{\text{init}} \rangle \tag{4.31}$$

We introduce the quantity $\mathcal{G}_{x,x'} = \langle x|(\lambda I - H)|x'\rangle \frac{\langle x'|\Psi_{\text{init}}\rangle}{\langle x|\Psi_{\text{init}}\rangle}$ which is close to a Green's function supplemented with a guiding function $\Psi_{\text{init}}(x)$. Rearranging the terms in the equation above, we obtain:

$$Z = \sum_{x_0,\ldots,x_p} \mathcal{G}_{x_0,x_1} \cdots \mathcal{G}_{x_{p-1},x_p} |\Psi_{\text{init}}(x_0)|^2 \equiv \sum_{x_0,\ldots,x_p} W(x_0,\ldots,x_p) \tag{4.32}$$

Note that to ensure that the relation above can be interpreted as a classical partition function, $W(x_0,\ldots,x_p)$ hence all the terms $\mathcal{G}_{x,x'}$ should be positive. Whenever any of the $\mathcal{G}_{x,x'}$ is negative, we are faced with the famous sign problem.

Ground-state energy and calculation of other observables. Given that $W(x_0,\ldots,x_p)/Z$ can be interpreted as a probability distribution (granted there is no sign problem), let us see now how to compute the ground-state energy and other physical observables of interest. Let us consider the energy expectation value of the projected state $(\lambda I - H)^{p/2}|\Psi_{\text{init}}\rangle$:

$$\langle H\rangle = \frac{\langle\Psi_{\text{init}}|(\lambda I - H)^{p/2} H (\lambda I - H)^{p/2}|\Psi_{\text{init}}\rangle}{\langle\Psi_{\text{init}}|(\lambda I - H)^{p}|\Psi_{\text{init}}\rangle} = \frac{1}{Z}\sum_{x_0,\ldots,x_p} e_L(x_0) W(x_0,\ldots,x_p) \tag{4.33}$$

where we used the fact that H commutes with $(\lambda I - H)^{p/2}$ and $e_L(x_0)$ is the local energy as defined in Eq. 4.13. For diagonal operators $\mathcal{O}$, similarly we have that :

$$\langle\mathcal{O}\rangle = \frac{\langle\Psi_{\text{init}}|(\lambda I - H)^{q} \mathcal{O} (\lambda I - H)^{p-q}|\Psi_{\text{init}}\rangle}{\langle\Psi_{\text{init}}|(\lambda I - H)^{p}|\Psi_{\text{init}}\rangle} = \frac{1}{Z}\sum_{x_0,\ldots,x_p} \mathcal{O}(x_{p-q}) W(x_0,\ldots,x_p) \tag{4.34}$$

where we can choose $q = \lfloor p/2 \rfloor$. As we will see in next paragraph, it is possible to sample efficiently the probability distribution $\mathcal{P}(x_0,\ldots,x_p) = W(x_0,\ldots,x_p)/Z$ which makes possible the statistical estimation of observables by sampling the middle configuration of the snake $\mathcal{R} \equiv (x_0,\ldots,x_p)$ for diagonal operators or its end points for the energy.

Sampling $W(x_0,\ldots,x_p)$. Sampling is done by generating a sequence of reptiles $\mathcal{R}$ in a Monte Carlo Markov chain. Two basic reptile moves were introduced in [Baroni & Moroni 1999] and consist in shifting the snake to the right or the left as shown in Fig. 4.3.

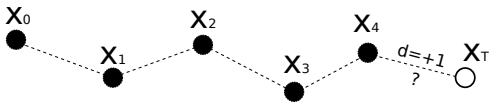


Figure 4.3: A reptile of size $p = 4$ and a right move $d = +1$ extending the snake with the configuration x_T.

We need to introduce the quantities $b_x = \sum_{x'} \mathcal{G}_{x,x'}$ and $p_{x',x} = \mathcal{G}_{x,x'}/b_x$ (giving $\sum_{x'} p_{x',x} = 1$) to write down the transition $t^{(d)}$ and acceptance $a^{(d)}$ probabilities for the right $(d = +1)$ and left $(d = -1)$ moves:

$$t^{(+1)}(\mathcal{R}'|\mathcal{R}) = p_{x_T,x_0} \quad , \quad a^{(+1)}(\mathcal{R}'|\mathcal{R}) = \min\left[1, \frac{b_{x_0}}{b_{x_{p-1}}}\right] \tag{4.35}$$

$$t^{(-1)}(\mathcal{R}'|\mathcal{R}) = p_{x_T,x_p} \quad , \quad a^{(-1)}(\mathcal{R}'|\mathcal{R}) = \min\left[1, \frac{b_{x_p}}{b_{x_1}}\right] \tag{4.36}$$

What is important to notice in the above equations is that selection and acceptance only needs values of b_x and $p_{x,x'}$ from either ends of the reptile. Moreover, it solves the low acceptance problem occurring in Green's function Monte Carlo. There, acceptance is computed as a product of $b_x \cdots b'_x$ weights whose number of terms grows along sampling and eventually vanishes[Becca & Sorella 2017]. We refer the interested reader to [Becca & Sorella 2017] for the detailed derivation of these relations and the proof that it satisfies detailed balance. A sampling scheme called the bounce algorithm was shown to decrease the autocorrelation between samples[Pierleoni & Ceperley 2005]. All in all, for sufficiently large p, the ground-state energy and diagonal observables can be estimated from N_{samples} independent samples $\{\mathcal{R}_i\}$:

$$E_0 \approx \frac{1}{N_{\text{samples}}} \sum_{\substack{i=0 \\ \mathcal{R}_i=(x_0,\ldots,x_p)}}^{N_{\text{samples}}} \frac{e_L(x_0)+e_L(x_p)}{2} \tag{4.37}$$

$$\langle \mathcal{O} \rangle \approx \frac{1}{N_{\text{samples}}} \sum_{\substack{i=0 \\ \mathcal{R}_i=(x_0,\ldots,x_p)}}^{N_{\text{samples}}} \mathcal{O}(x_{\lfloor p/2 \rfloor}) \tag{4.38}$$

where we used also in Eq. 4.37 the equivalence of the endpoints of the snake.

4.3.2 RBM as guiding wave functions

When $|\Psi_{\text{init}}\rangle$ is chosen to be a more educated guess than just a random state, it is called a guiding wave function. If $|\Psi_{\text{init}}\rangle$ is close to the exact ground-state, i.e. $a_i/a_0 \ll 1$ in Eq. 4.29, we naturally expect faster convergence (still exponential but with a better prefactor), therefore reaching a certain error threshold could require less projections. Another important advantage of the guiding wave function is the fact that it can cure the sign problem in some cases. Indeed, if the Green's function $\langle x|(\lambda I - H)|x'\rangle$ is negative for some configurations x and x', the sign structure of H can sometimes be cancelled out by encoding it in $|\Psi_{\text{init}}\rangle$, hence ensuring $\mathcal{G}_{x,x'} > 0$.

We refer to Sec. A.2 of appendix A for details on the complexity of the RQMC algorithm. The dominant cost of to obtain N_{samples} samples with p projections is $O(N \times p \times N_{\text{samples}})$ without a guide and $O(\alpha N^2 \times p \times N_{\text{samples}})$ for both $M \equiv \alpha N$ non-symmetric RBMs and $M \equiv \alpha$ symmetric RBMs.

4.4 Study of a two-dimensional ring-exchange model

The prototypical ground state of a fermionic system is a Fermi liquid where electrons lie inside a sphere in momentum space called the Fermi surface. Low-energy excitations happen only near this surface because of the Pauli principle. Bosonic systems typically form Bose-Einstein condensates (BEC) at $T = 0$ where all bosons behave as a whole in the form of a coherent macroscopic quantum wave function. When the repulsion is increased between bosons in a lattice, superfluidity arising in BEC is often replaced by a Mott insulating phase in which bosons localize in a crystalline arrangement. In [Paramekanti *et al.* 2002], Paramekanti, Balents and Fisher proposed the existence of a new bosonic state of matter sharing many similarities with fermionic states. They provide theoretical arguments in favor of the stability of this phase if ring-exchange kinetic terms (see Fig. 4.4) are sufficiently strong compared to usual near-site

hopping. These findings were particularly exciting as it was argued to be relevant in the context of solid 3-He[Roger 1983] and possibly related to high-T_c superconductivity. Alternatively, one can view ring-exchange processes as the hopping of a pair of particles – one boson and one hole forming an *exciton* – throughout the lattice, thus motivating the name *exciton Bose liquid* (EBL). EBL features the following thermodynamic properties: finite compressibility, specific heat in $\sim T\log(T)$ and the existence of a 1D manifold of gapless excitations, which is reminiscent of liquids[Paramekanti *et al.* 2002].

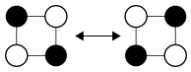


Figure 4.4: A ring-exchange moves two bosons located on the diagonal of a plaquette to the opposite diagonal.

4.4.1 The $K_1 - K_2$ model

Following up on a series of attempts[Sandvik *et al.* 2002, Melko *et al.* 2004, Rousseau *et al.* 2004, Rousseau *et al.* 2005] to realize the EBL phase in Hamiltonian models, Tay and Motrunich [Tay & Motrunich 2010, Tay & Motrunich 2011] studied a model of hardcore bosons with two types of ring-exchange moves where they provide numerical evidences in favor of the existence of this phase. Let us first define ring-exchange operators on the square lattice on plaquette $\mathbf{r}$:

$$P^{mn}_{\mathbf{r}} = b^\dagger_{\mathbf{r}} b_{\mathbf{r}+m\hat{\mathbf{x}}} b^\dagger_{\mathbf{r}+m\hat{\mathbf{x}}+n\hat{\mathbf{y}}} b_{\mathbf{r}+n\hat{\mathbf{y}}} \tag{4.39}$$

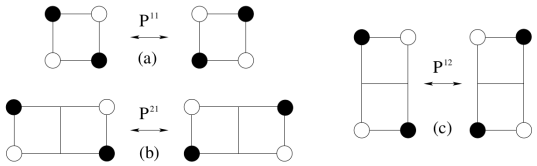


Figure 4.5: The ring exchange operators for (a) 1×1, (b) 2×1 and (c) 1×2 plaquettes. Figure from [Tay & Motrunich 2010].

Fig. 4.5 shows the three types of plaquette operators appearing in the Hamiltonian considered in [Tay & Motrunich 2010, Tay & Motrunich 2011] which reads:

$$H = -K_1 \sum_{\mathbf{r}} P^{1\times 1}_{\mathbf{r}} - K_2 \sum_{\mathbf{r}} \left(P^{1\times 2}_{\mathbf{r}} + P^{2\times 1}_{\mathbf{r}}\right) \tag{4.40}$$

Without loss of generality, we set $K_1 = 1$ and vary $K_2 \geq 0$ in the following. The model with $K_2 = 0$ was considered earlier in [Melko *et al.* 2004] and was shown to exhibit a (π,π) charge-density wave (CDW) phase. This could be expected since energy decreases as more 1×1 plaquettes are hoppable, the ground state will lie in the sector containing the Néel boson configuration (see Fig. 4.7a where all the L^2 plaquettes of the lattice are hoppable, L being the linear size of the square lattice). It was shown in [Tay & Motrunich 2011] that the EBL phase is unstable to

such a (π, π)-CDW due to the presence of Umklapp terms that destroy the liquid. Model (4.40) can be considered as the simplest model including only ring-exchange terms that is capable of destabilizing a CDW order due to the presence of the K_2 terms. Indeed, the K_2 terms do not act on the Néel configuration, they compete with K_1 terms and also between each other. In addition to that, extended ring-exchanges have the effect of mitigating phase separation tendencies as observed away from half-filling[Rousseau *et al.* 2004, Rousseau *et al.* 2005]. Despite the absence of a sign problem for non-negative K_1 and K_2 (all terms in Hamiltonian (4.40) are negative), the model is hard to simulate with QMC. Indeed, efficient update schemes are hard to design due to the ring-exchange term[Melko & Sandvik 2005]. As a consequence, Tay and Motrunich used variational and Green function Monte Carlo[Calandra-Buonaura & Sorella 1998] methods to obtain the phase diagram in Fig. 4.6.

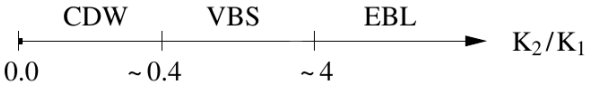


Figure 4.6: Phase diagram for model 4.40 as obtained in [Tay & Motrunich 2011]. A charge-density wave (CDW) phase develops at small K_2/K_1, then a columnar valence-bond state (VBS) was observed at moderate values of K_2/K_1, then the exciton Bose liquid phase starts off beyond $K_2/K_1 \approx 4$.

The relevant order parameters or observables of each phase are listed below:

1. **Density structure factor – the CDW phase.** An important observable to detect ordering is the density structure factor, it is defined as the Fourier transform of density-density correlations:

$$S(q_x, q_y) = \frac{1}{L^2} \sum_{\mathbf{r},\mathbf{r}'} e^{i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')} \langle n_{\mathbf{r}} n_{\mathbf{r}'} - \bar{n} \rangle \tag{4.41}$$

   A natural order parameter for a (π, π)-CDW is the value of the density structure factor at $\mathbf{q} = (\pi, \pi)$ where it reaches a maximum for a staggered boson occupation (Neéel state). The value of $S(\pi, \pi)$ can also be written as:

$$S(\pi, \pi) = \left\langle \left| \frac{1}{L} \sum_{\mathbf{r}=(r_x,r_y)} (-1)^{r_x+r_y} (n_{\mathbf{r}} - \bar{n}) \right|^2 \right\rangle \tag{4.42}$$

   whose computational cost scales like N rather than N^2 if calculated naively from Eq. 4.41.

2. **Plaquette structure factor – the VBS phase.** Likewise, the corresponding order parameter for a $(0, \pi)$ valence bond solid (VBS) is obtained from plaquette structure factor defined as follows:

$$P(q_x, q_y) = \frac{1}{L^2} \sum_{\mathbf{r},\mathbf{r}'} e^{i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')} \left\langle \left(P^{11}_{\mathbf{r}}\right)^2 \left(P^{11}_{\mathbf{r}'}\right)^2 \right\rangle \tag{4.43}$$

   whose value at $\mathbf{q} = (0, \pi)$ can be used as an order parameter of this phase. Again, it is possible to obtain a similar simplified expression as in Eq. 4.42.

3. **Cross observable – the EBL phase.** The conditions for the existence of an EBL phase were derived in [Paramekanti *et al.* 2002]: (i) there should not be any charge ordering, which can be probed by the existence/absence of Bragg peaks in the Brillouin zone and (ii) the density structure factor should display singular lines $(0, q_y)$ and $(q_x, 0)$ which are theoretically identified with the presence of a Bose surface. The second condition can be examined based on the cross operator defined in [Tay & Motrunich 2011]:

$$\sigma(q_x, q_y) = \frac{S(q_x, q_y)}{4|\sin(q_x/2)\sin(q_y/2)|} \tag{4.44}$$

where $S_{\mathrm{EBL}}(q_x, q_y) \equiv 4|\sin(q_x/2)\sin(q_y/2)|$ is the prediction of the EBL theory [Paramekanti *et al.* 2002]. As argued in [Tay & Motrunich 2011], a positive identification of the EBL phase is signaled if the value of σ near the singular lines stays finite in the thermodynamic limit since in that case $S(q_x, q_y) \underset{q_x \to 0}{\propto} S_{\mathrm{EBL}}(q_x, q_y)$ (or equivalently when $q_y \to 0$). Moreover, [Tay & Motrunich 2011] argued based on an effective field theory that in order for the EBL phase to be stable, σ should stay above a threshold value $\sigma \geq \sigma_c = 3/16$. In practice, we will evaluate $\sigma(q_{\mathrm{min}} = 2\pi/L, q_y)$ or equivalently $\sigma(q_x, q_{\mathrm{min}} = 2\pi/L)$.

4.4.2 Conservation laws and consequences

The peculiarity of the $K_1 - K_2$ model stems from the presence of ring-only kinetic terms which induces a number of symmetries and associated conservation laws. Hamiltonian (4.40) conserves the total number of bosons ($U(1)$ symmetry), it is particle-hole symmetric and inherits the spatial symmetries of the square lattice (i.e. invariant under the action of the C_{4v} point-group corresponding to the L^2 translations and 8 rotations and reflections of the lattice). More importantly, *the number of bosons per line and per column is conserved* (associated to $2L$ symmetries). These latter conservation laws have profound consequences on the properties of the system such as highly constrained dynamics since the bosons can only move in a correlated manner. Another direct consequence is the existence of a zero-energy manifold along the lines $(q_x, 0)$ and $(0, q_y)$ [Paramekanti *et al.* 2002] from which the Bose surface of low-energy gapless excitations originates.

The existence of these conservation laws means that the Hilbert space will fragmentate in as many sectors as specific values of the conserved quantities. In the following, we will study the half-filled lattice, i.e. $\bar{n} = \frac{1}{2}$ sector, and concentrate on the sector of half-filled columns and rows. We checked for the smallest system sizes that the ground-state is in that sector and its size has a favorable scaling $2^{L^2 - L\log(L) + L(\log(2) - \log(\pi)) + o(L)}$ (that we derived from the results of [Canfield & McKay 2005]). Since the variational and projection methods rely on sampling boson configurations, one has to verify whether sampling can be ergodic if new configurations are obtained from a Hamiltonian move (i.e. boson configurations are transformed according to 1×1, 1×2 or 2×1 plaquette ring-exchanges). Clearly, sampling preserves the same conserved quantities as the Hamiltonian, that is the number of bosons and the number of bosons per rows and columns. However, we noticed that some configurations of bosons are kinetically frozen as can be seen Fig. 4.7b, meaning that they are connected to no other configurations by an element of the Hamiltonian. For $K_1 \neq 0$ and $K_2 \neq 0$, we checked that they represent a negligible portion of the sector size: 0 for $L = 4$, 12 for $L = 6$ (the exact enumeration for larger system sizes is

too expensive). For $K_1 = 0$ or $K_2 = 0$, more care has to be taken as we noticed fragmentation of the Hilbert space into an extensive number of Krylov subspaces similar to what was observed in [Khemani & Nandkishore 2020, Moudgalya *et al.* 2019, Sala *et al.* 2020] but this is out of the scope of our current study and we leave this for future investigations.

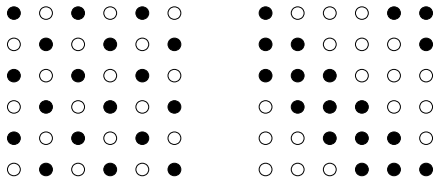


Figure 4.7: (Left) Néel state where all 1×1 plaquette of the lattice are ring-exchangeable. (Right) Example of a kinetically frozen configuration for the $K_1 - K_2$ model on a 6×6 lattice. This boson configuration (bosons are black filled circles, holes are white circles) is in the $\bar{n} = \frac{1}{2}$ sector with half-filled rows and columns but is connected to no other configurations by the dynamics produced by the Hamiltonian, i.e. $H_{c_f,c'} = 0$ for all other configuration c' of the sector with c_f the frozen configuration. For $L = 6$, there are 12 such frozen states related by 6 x-translation and a reflection.

Motivated by the burst of activity surrounding NQS, our work is an attempt to extend the findings of [Tay & Motrunich 2010], i.e. confirm or contradict their results on larger system sizes (up to $L \leq 16$ compared to $L = 12$ in [Tay & Motrunich 2010]) with a variational study using NQS and guided reptation QMC methods as presented in previous sections. This also constitutes an interesting benchmark to test the efficiency of NQS in this multifaceted phase diagram.

4.4.3 VMC results

Although RBM wave functions are provably unable to approximate efficiently certain many-body states, they achieved excellent performance in many different contexts and stand as one of the simplest NQS to use. A key ingredient in the success of variational approaches is the implementation of symmetries of the model directly into the variational ansatz. Here, given that $-H$ is real and non-negative in the basis of boson occupation number, the Perron-Frobenius theorem ensures that the ground-state is unique and can be written as a real and positive vector, thus we will consider RBM with only real parameters. The uniqueness of the ground state proves that it possesses the same symmetries of the Hamiltonian[2]. Throughout our variational study, we will use three types of RBM ansatz, each being invariant to different symmetries of model (4.40):

- a *real RBM* (denoted rRBM hereafter) based on the non-symmetric ansatz of Eq. 4.3 with $M \equiv \alpha N$ hidden nodes which amounts to $\alpha N^2 + (\alpha + 1)N$ variational parameters.
- a *real translation-invariant RBM* (denoted $\mathcal{T}$-invariant rRBM) based on the symmetric

[2]Since for any eigenstate $|E\rangle$ and a symmetry U of the Hamiltonian, $U|E\rangle$ is also an eigenstate of the Hamiltonian with the same energy E.

ansatz of Eq. 4.4 implementing the N lattice translations. We choose a number of filters $M \equiv \alpha$, thus this ansatz includes $(\alpha+1)N+\alpha$ variational parameters.

- a *real C_{4v}-invariant RBM* (denoted hereafter C_{4v}-invariant rRBM) containing the symmetries of the $\mathcal{T}$-invariant rRBM supplemented with the 8 rotations and reflections of the lattice, thereby enforcing the complete C_{4v} point-group symmetry of the square lattice. It includes the same amount of variational parameters as the $\mathcal{T}$-invariant RBM but results in a computational overhead proportional to the number of additional symmetries ($\times 8$).

Setting the visible biases a_i to 0 is a sufficient condition to enforce particle-hole symmetry, however we did see only little improvement of the performance doing this. Sampling according to Hamiltonian moves allows to conserve the number of bosons per row and column during the sampling procedure, which also preserves the $U(1)$ symmetry. We empirically chose the hyperparameters of the simulation. In most runs, the learning rate was set around 0.05 but needed to be decreased for larger values of K_2 and larger lattices. We noticed that stochastic reconfiguration produces more stable learning and faster convergence (in terms of the number of VMC iterations) than simple gradient descent. The VMC calculations were done with the Netket library [Carleo *et al.* 2019a].

In the following, we start by benchmarking the RBM ability to accurately approximate the ground state of model (4.40) by comparing the energy and the value of other observables of optimized RBMs to exact diagonalization for $L = 6$ (maximum achievable) and to the Green's function Monte Carlo results of [Tay & Motrunich 2011] for $L = 12$. After that, we attempt a finite-size scaling analysis of the different order parameters to map out the phase diagram.

4.4.3.1 Benchmark against exact diagonalization on $L = 6$

Fig. 4.8 shows the accuracy of the observables of interest as a function of the ansatz complexity α, in the three regimes revealed by [Tay & Motrunich 2010]: $K_2 = 0$ representative of the CDW phase, $K_2 = 1$ representative of the VBS phase and $K_2 = 7$ representative of the EBL phase. First of all, we note that the precision of all observables is better than the one-parameter variational ansatz of [Tay & Motrunich 2011] that was based on spin wave theory, this holds for all K_2 regimes and all RBM ansatz considered. Energy and fidelity can be systematically improved by increasing α, we can reach 10^{-3} relative error with moderate effort from $\alpha = 4$ on, also the precision has not yet saturated for the symmetric ansatz for large α and large K_2 which is reassuring regarding the expressive power of the RBMs. As K_2 is increased, the relative error slightly increases which can be explained by the fact that the large-K_2 regime is gapless (see Eq. 4.11). Finally, the advantage of implementing symmetries is more visible for $K_2 \leq 1$ and large α, as can be seen from the fact that the relative error on the energy and fidelity of rRBMs saturates at higher values than with $\mathcal{T}$- or C_{4v}-invariant rRBMs.

The relative error on $S(\pi,\pi)$ and $P(0,\pi)$ is not a clear monotonously decreasing function of α since these observables are not explicitly optimized (although $P(0,\pi)$ behaves somewhat better). It is in fact known that the error on these observables scales as $\sqrt{1-\langle E_0|\Psi_{\mathrm{RBM}}\rangle}$ (see [Becca & Sorella 2017]), which roughly applies here (10^{-3} precision on fidelity translates into $\sim 10^{-2}$ for $S(\pi,\pi)$ and $P(0,\pi)$). We checked that these results hold for other values of K_2.

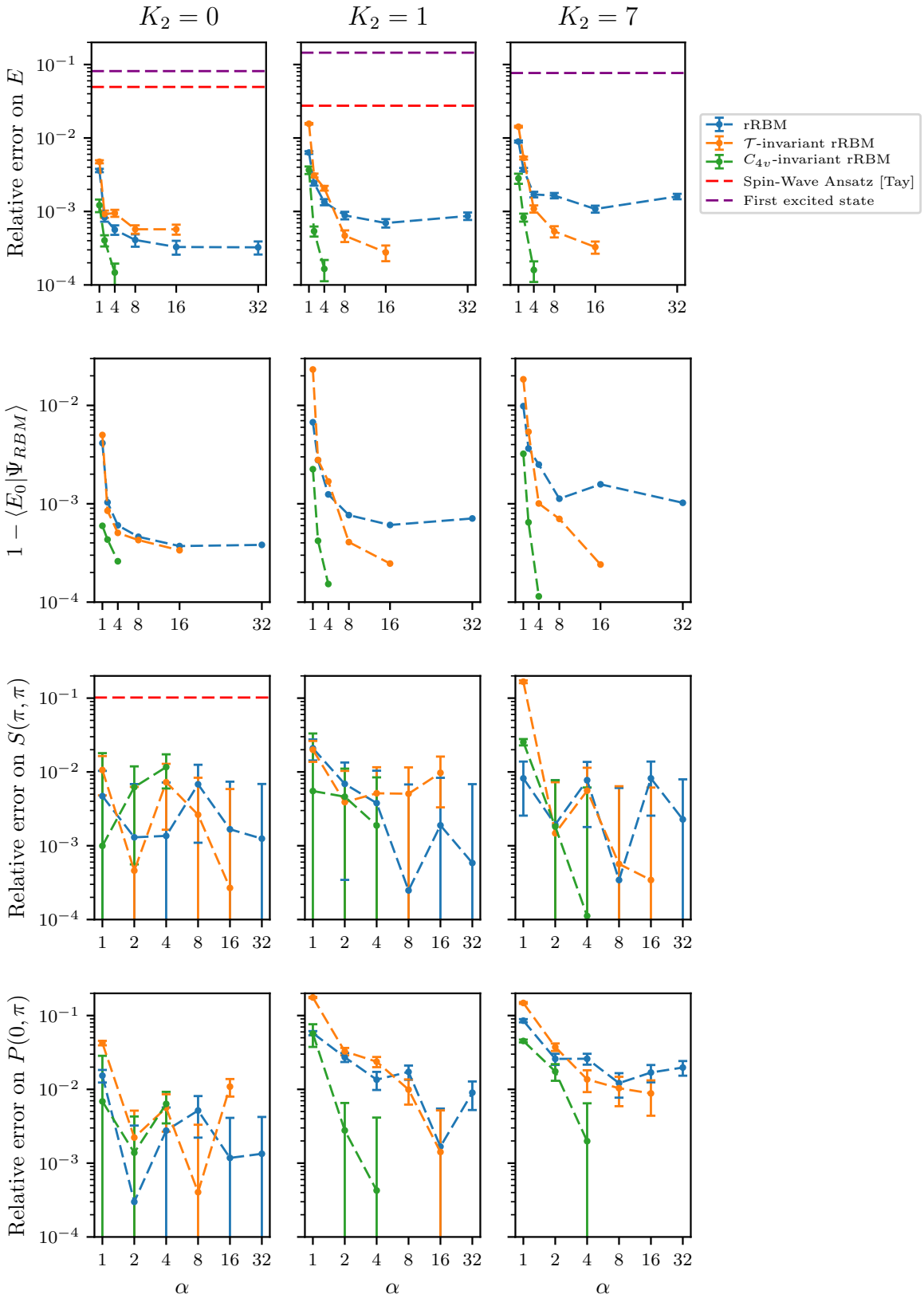


Figure 4.8: Relative error on (a) the energy, (b) overlap $1 - \langle E_0 | \Psi_{\mathrm{RBM}} \rangle$, (c) $S(\pi, \pi)$, (d) $P(0, \pi)$, with respect to the exact ground-state for $L = 6$ at different values of K_2 as a function of the RBM complexity α. The relative error on each observable $\mathcal{O}$ is computed as $|\mathcal{O}^{\mathrm{RBM}} - \mathcal{O}^{\mathrm{exact}}|/|\mathcal{O}^{\mathrm{exact}}|$. For clarity purposes, we set the α axis as logarithmic for $S(\pi, \pi)$ and $P(0, \pi)$. The variational results of [Tay & Motrunich 2011] are represented as red dashed lines. The energy of the first-excited state is represented as purple dashed lines. All observables are evaluated on 50000 configurations sampled from the corresponding RBMs.

4.4.3.2 Benchmark against GFMC on $L = 12$

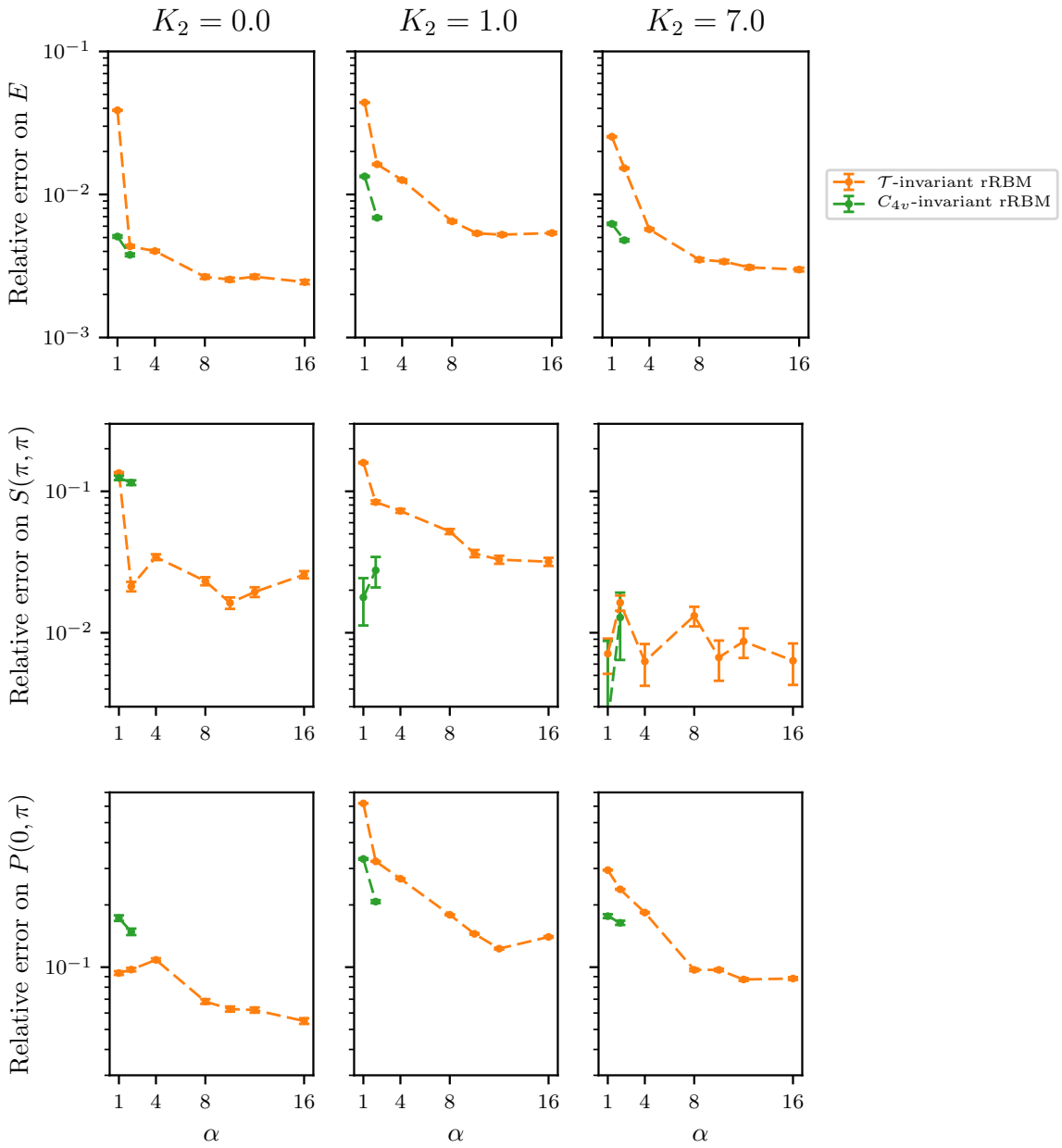


Figure 4.9: Relative error of observables with respect to the best RQMC results (see Sec. 4.4.4), i.e. $|\mathcal{O}^{\mathrm{RBM}} - \mathcal{O}^{\mathrm{RQMC}}|/|\mathcal{O}^{\mathrm{RQMC}}|$, obtained from different RBMs trained at different K_2 and $L = 12$ plotted as a function α. (Top) Energy. (Middle) Density structure factor $S(\pi, \pi)$. (Bottom) Plaquette structure factor $P(0, \pi)$.

We benchmark our variational wave functions on $L = 12$ against the best RQMC results of Sec. 4.4.4 which can be considered exact. Here we discard rRBMs since they displayed the worst performance in last section. The precision on the energy is slightly worse (by a factor of 10) than for $L = 6$ (see Fig. 4.8) but remains good. The precision stabilizes at large α and shows

the inherent approximation bias of the $\mathcal{T}$-invariant rRBMs.

Given the VMC calculations are 8 times slower for C_{4v}-invariant RBMs than $\mathcal{T}$-invariant RBMs for a given α (see Sec. A.1 of appendix A), the largest α data points represented in Fig. 4.9 ($\alpha = 2$ for C_{4v}-invariant RBMs and $\alpha = 16$ for $\mathcal{T}$-invariant RBMs) in fact carry the same computational cost. Knowing this, the precision of the C_{4v}-invariant ansatz is quite disappointing since it has a larger relative error on all observables and in all regimes (except $S(\pi,\pi)$ at $K_2 = 1$) when compared to $\mathcal{T}$-invariant RBMs of the same computational cost.

4.4.3.3 Variational phase diagram

Let us establish the phase diagram of the $K_1 - K_2$ model from our RBM-VMC setup. In particular, we are interested in seeing whether the RBMs are able to reveal the plaquette ordered phase that was missed by the variational ansatz of [Tay & Motrunich 2011]. We perform a finite-size scaling analysis of the CDW and VBS order parameters calculated from $\mathcal{T}$-invariant rRBM wave functions with $\alpha = 10$ across the whole phase diagram for system sizes up to $L = 16$. The scaling is done on the quantities $S(\pi,\pi)/L^2$ and $P(0,\pi)/L^2$, which we expect to converge to a finite non-zero value in the thermodynamic limit. We also looked at their respective Binder cumulant that is defined as follows for an observable $\mathcal{O}$:

$$\text{Binder ratio} = \frac{\langle|\mathcal{O}|^4\rangle}{\langle|\mathcal{O}|^2\rangle^2} \tag{4.45}$$

The advantage of such a quantity is that it is expected to approach 1 in the presence of order and 3 in the absence of order, moreover it is less sensitive to finite-size effects.

As visible in Fig. 4.10, $S(\pi,\pi)/L^2$ does not renormalize down to 0 for $K_2 \leq 0.4$ and it is hard to conclude for larger K_2. The Binder cumulant is more instructive as it clearly reaches 3 for all system sizes around $K_2 = 2$ which means there is no (π,π)-CDW order beyond this value. For smaller $K_2 \leq 1$, although its value is close to 3, the Binder cumulant seems to slowly renormalize downwards, which does not strictly rule out the possibility of order up until $K_2 = 1$.

Again, it is hard to tell whether $P(0,\pi)/L^2$ vanishes or not for large L, it is possible to see that this quantity decreases faster with L for $K_2 \lesssim 0.3$ (red/orange curves) than for higher K_2 around $K_2 = 0.6$ (green curves). The trends can be more easily analyzed from the Binder cumulant: an upward renormalization at low $K_2 \leq 0.4$ indicates the instability of a VBS in this low K_2 regime. The Binder cumulant clearly shows the absence of order beyond $K_2 \geq 2$ where its value exceeds 3 for all system sizes. For intermediate $0.5 \leq K_2 < 2$, plaquette correlations are well captured by the RBM wave function and we see downwards finite-size trends which suggests the stability of a plaquette-ordered phase.

For $K_2 = 1.0$, the EBL diagostics σ clearly renormalizes down to zero which signals the absence of an EBL phase. Almost all the values of σ are below the threshold $\sigma_c = 3/16$ for $K_2 = 3.0$ which sets a lower bound for the extension of the EBL phase at small K_2. The criterion is almost not violated for $K_2 = 7.0$, only for large momentum close to π. As acknowledged in [Tay & Motrunich 2010], the stability threshold may not in fact be considered as strict. Given the biased nature of VMC and the imperfection of the stability criterion, we do not conclude on the existence of the EBL phase for the moment and we will re-examine our results in Sec. 4.4.4 in light of the exact RQMC results.

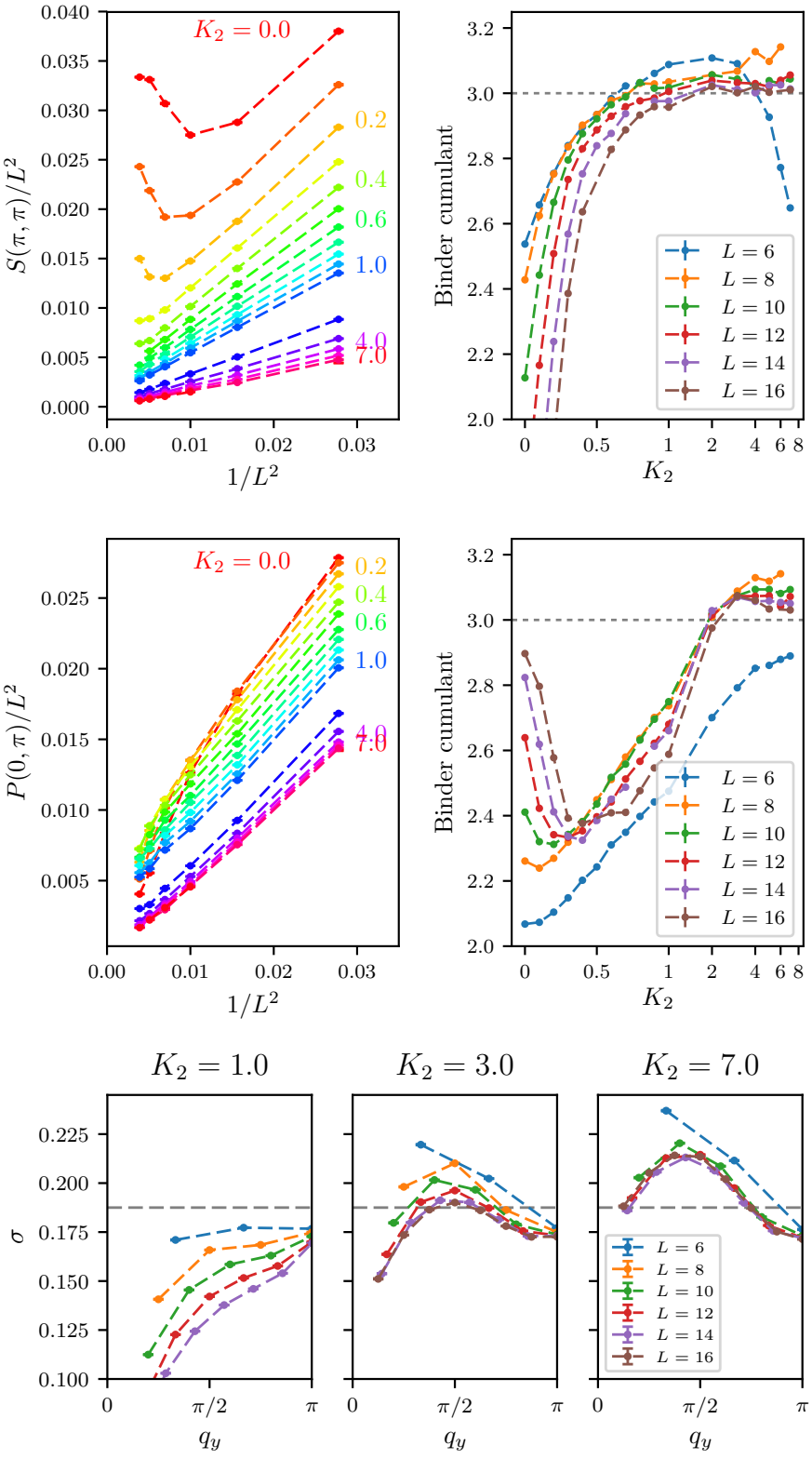


Figure 4.10: (Top left) $S(\pi,\pi)/L^2$ plotted against L^2 for different values of K_2. (Top right) Binder cumulant of $S(\pi,\pi)$ for system sizes from $L = 6$ to $L = 16$. Note that we use a linear scale in the range $K_2 \in [0,1]$ and a log scale for $K_2 \in [1,10]$. (Middle left) $P(0,\pi)/L^2$ plotted against L^2 for different values of K_2. (Middle right) Binder cumulant of $P(0,\pi)$ for system sizes from $L = 6$ to $L = 16$. (Bottom) Cross operator $\sigma(2\pi/L, q_y)$ (as defined in Eq. 4.44) plotted against q_y for different values of K_2 and system sizes up to $L = 16$. The dashed line indicates the critical value of $\sigma_c = 3/16$ below which the EBL phase is not stable.

The VMC results obtained with a $\mathcal{T}$-invariant rRBM wave function with $\alpha = 10$ are summarized in Fig. 4.11. We obtain good qualitative agreement with the phase diagram obtained in [Tay & Motrunich 2010] with GFMC.

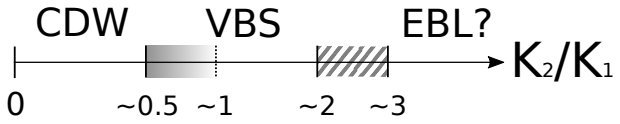


Figure 4.11: VMC phase diagram obtained with $\mathcal{T}$-invariant rRBMs with $\alpha = 10$ from finite-size scaling on systems up to $L = 16$. The first color gradient indicates the conflicting coexistence of the CDW and VBS phases. The dashed area indicates a region where we detect no plaquette correlations and where the EBL is clearly unstable based on our criterion.

4.4.4 RQMC results

Last section showed that even though our purely variational approach with a RBM ansatz is quite accurate on $L = 6$ and on $L = 12$, it was sometimes difficult to establish a phase diagram with a high degree of uncertainty given the too slow or inhomogeneous finite-size trends of the order parameters. Also more fundamentally, this remained a variational study. In this section, we turn to results we obtained using a reptation QMC approach. This method fundamentally differs from variational methods in that it is *unbiased*. As pointed out in Sec. 4.3.1, using a good starting wave function (close to the actual ground-state wave function) allows to speed up the convergence as well as providing statistical estimations with less variance. As a result, we will compare unguided projections with projections guided with the RBM wave functions optimized in last section. To do so, we implemented the RQMC algorithm on top of the Netket library [Carleo *et al.* 2019a] to be able to use RBMs as guiding wave functions.

4.4.4.1 Benchmark against exact diagonalization on $L = 6$

In Fig. 4.12, we benchmark RQMC results against exact diagonalization and in particular compare the performance of unguided projections compared to ones with a $\mathcal{T}$-invariant rRBM wave function as a guide. We did not include non-symmetric rRBMs in our analysis because this ansatz carries the same computational cost as $\mathcal{T}$-invariant rRBMs (see Sec. A.2 in appendix A) while having no symmetry implemented. Likewise we do not include C_{4v}-invariant rRBMs due to their prohibitive computational cost (8 times slower than $\mathcal{T}$-invariant rRBMs), we will give further justification in next section on $L = 12$ simulations.

For unguided RQMC, the energy converges in ~ 40 projections up to an error of 10^{-4} for all K_2, which is slightly better than our best variational results. Using a guide clearly speeds up convergence since the energy is converged in 20 projections (sometimes less) for all K_2 and α. We are able to reach accuracies of the order of 10^{-5} for the energy and 10^{-3} for the observables, although we note that the unguided results include the best guided results within its error bars.

However, to make a fair comparison between unguided and guided projections, one has to take into account the additional computational cost that comes with evaluating the ratio $\Psi(x')/\Psi(x)$ with a guide. As explained in Sec. 4.3.1, the cost is proportional to α. Additionally, we defined a Monte Carlo sweep as p local moves, which means the computational time is also

proportional to p. An important quantity is the autocorrelation time τ, which in essence tells that the samples used for the statistical estimation should be picked every τ steps in the Markov chain. The bigger τ, the more samples we need to obtain a good statistical uncertainty. The bottom figure in Fig. 4.12 shows the relative energy error as a function of the time needed to obtain one uncorrelated sample, which is proportional to α, p and τ as explained above. Then, the most time-efficient ansatz would be the one that can reach low error with minimum time (therefore it is best if the points accumulate in bottom left corner of Fig. 4.12d). Unfortunately, it is quite difficult to conclude from Fig. 4.12d which setup is the most time-efficient: indeed for $K_2 = 0$ or $K_2 = 7$, $\alpha = 1$ seems best but $\alpha = 4$ seems preferable for $K_2 = 1$. We will reexamine these results for $L = 12$ below.

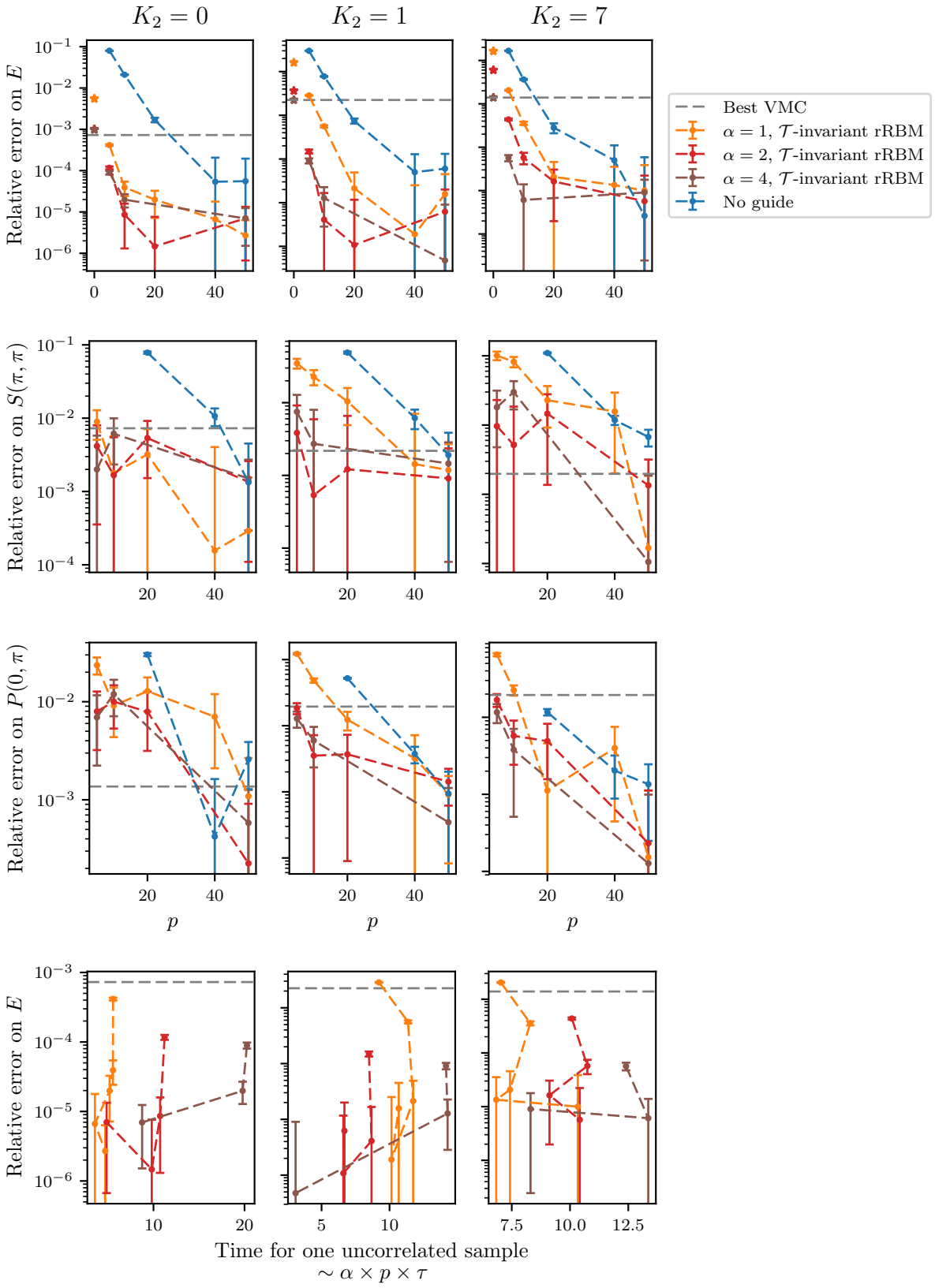


Figure 4.12: Relative error on (a) the energy, (b) $S(\pi, \pi)$, (c) $P(0, \pi)$ with respect to the exact ground-state for $L = 6$ at different values of K_2 as a function of the number of projections. Our best variational results are showed in grey dashed lines. All observables are evaluated on 6.10^6 configurations. For the energy, the stars indicate the VMC energy of the respective guiding wave functions. The last row (d) shows the same data as in (a) but plotted against the time needed to get one uncorrelated sample which is proportional to $\alpha \times p \times \tau$ where τ is the autocorrelation time between samples.

4.4.4.2 Benchmark against GFMC on $L = 12$

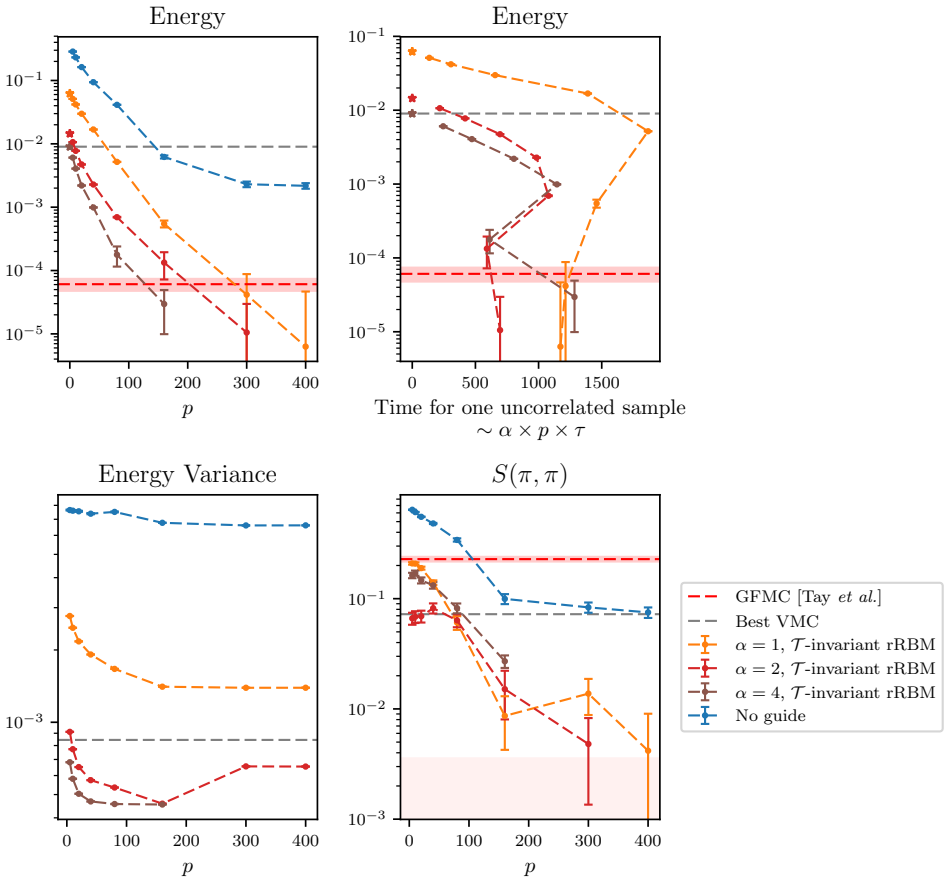


Figure 4.13: Observables obtained with RQMC without a guide (blue) or guided by $\mathcal{T}$-invariant rRBMs trained at $K_2 = 0.5$ and $L = 12$ plotted as a function the number of projections. (Top left) Energy relative error with respect to the best RQMC energy we obtained ($\alpha = 2$, $p = 400$). (Top right) Same energy data plotted against the time needed to compute one independent sample (see main text) (Bottom left) Relative energy variance $\langle (H - E)^2 \rangle / E^2$. (Bottom right) $S(\pi, \pi)$ relative error with respect to the best RQMC data ($\alpha = 2$, $p = 400$). The grey dashed (red dashed) lines indicate our best VMC results (GFMC results from [Tay & Motrunich 2011] with error bar shown with the shaded area). The red shaded area shows the error bar on $S(\pi, \pi)$ for the reference RQMC point ($\alpha = 2$, $p = 400$).

The exponential decrease of the relative energy error with p (line in log scale) confirms the exponential convergence expected from RQMC as pointed out in Sec. 4.3.1. The projections converge for a larger number of projections than $L = 6$ (more than 200 here), this is due to smaller relative energy gaps for larger systems. For unguided RQMC, the energy seemingly converge to a relative error of 10^{-3}, however we think this saturation may be interpreted as an artifact of very long equilibration we observed for the largest projections (the $p = 300$ and $p = 400$ data were likely not thermalized long enough). We did not notice such limitations for guided projections, which also show lower relative energy variance and better precision on

$S(\pi,\pi)$. As for $L=6$, the observables are more accurate when α and p are increased, the top right panel in Fig. 4.13 thus plots the relative energy error as a function the time needed to obtain one uncorrelated sample. Fig. 4.13 seems to suggest that the most time-efficient guide is a $\mathcal{T}$-invariant rRBM with $\alpha=2$, however the variable outcomes obtained from the same analysis on $L=6$ suggest that we should perform the same analysis for other values of K_2.

4.4.4.3 RQMC phase diagram

Fig. 4.14 summarizes the results we obtained with RQMC guided with $\mathcal{T}$-invariant rRBMs with $\alpha=1$, which were optimized for each K_2 and system size. We motivate the choice of $\alpha=1$ by the following arguments: (i) we mention that we did the previous benchmark and time efficiency analysis after obtaining the results of this section, (ii) ideally, *knowing the autocorrelation time before starting the Monte Carlo sampling*, one would generate just the amount of samples needed to obtain a given statistical uncertainty, however crucially, this is not possible, we found easier in practice to generate a large amount of samples (without fine-tuning) in order to have reasonably small error bars. As a consequence, this excluded the use of costly guiding wave functions like $\mathcal{T}$-invariant for $\alpha>4$ or C_{4v}-invariant rRBMs.

As is visible from the Binder cumulant of $S(\pi,\pi)$, the (π,π)-CDW order develops no further than $K_2\geq 0.5$ beyond which the Binder cumulant reaches the disordered value 3 and the finite-size curves overlap without a clear trend. The plaquette correlations $P(0,\pi)/L^2$ vanish in the CDW phase at low $K_2<0.5$, which is confirmed by the upward trend of the Binder cumulant for the largest system sizes. Based on the Binder cumulant, the VBS phase could extend up to $K_2\approx 3.5\sim 4$ where all the curves seem to overlap (showing no renormalization).

Regarding the EBL phase, for $K_2=3.8$, σ seems to renormalize down with system size but more importantly violates the stability criterion ($\sigma<3/16$) for almost all q_y values. For $K_2=7$, the situation is mixed, the downwards finite-size trends seem slower than for $K_2=3.8$ and $\sigma>3/16$ for a majority of the momenta q_y but not all. If the criterion is strictly followed, the latter observation would rule out the existence of the EBL phase. However, the derivation of the stability threshold is based on some assumptions made in [Tay & Motrunich 2011] that these authors acknowledge. The EBL is considered stable if the leading Umklapp term is irrelevant, which means its associated scaling dimension (Eq. A22 in [Tay & Motrunich 2011])

$$\Delta = 8\int_0^{\pi}\frac{\sigma(0,q_y)}{|C(0,q_y)|^2}\cos^2(q_y/2)\sin(q_y/2)dq_y \tag{4.46}$$

should be greater than 1. As argued in [Tay & Motrunich 2011], we can consider $|C(0,q_y)|^2=1$. To determine which values of $\sigma(0,q_y)$ satisfy this inequality, the authors of [Tay & Motrunich 2011] consider it to be constant, which allows to estimate the integral in Eq. 4.46. This leads to the criterion $\sigma>3/16$ to assess the stability of EBL. Nevertheless, given the function $x\mapsto\cos^2(x/2)\sin(x/2)$ is peaked around $\pi/2$ as is $\sigma(0,q_y)$ in our data, this criterion may be underestimating the stability of the EBL phase. As a further check that no other simpler order is present in this region of the phase diagram, we display the structure factor at $K_2=7$ for $L=16$ (Fig. 4.15) which does not display any Bragg peak. All in all, the $L=14$ and $L=16$ data at large K_2 (not available in [Tay & Motrunich 2010]) show that (i) the shape of the cross operator seems to stabilize on these system sizes and (ii) no order is visible for $L=16$, which further solidifies the conjecture of [Tay & Motrunich 2010] that the EBL is realized at large K_2 for model (4.40). The RQMC results are summarized in Fig. 4.16.

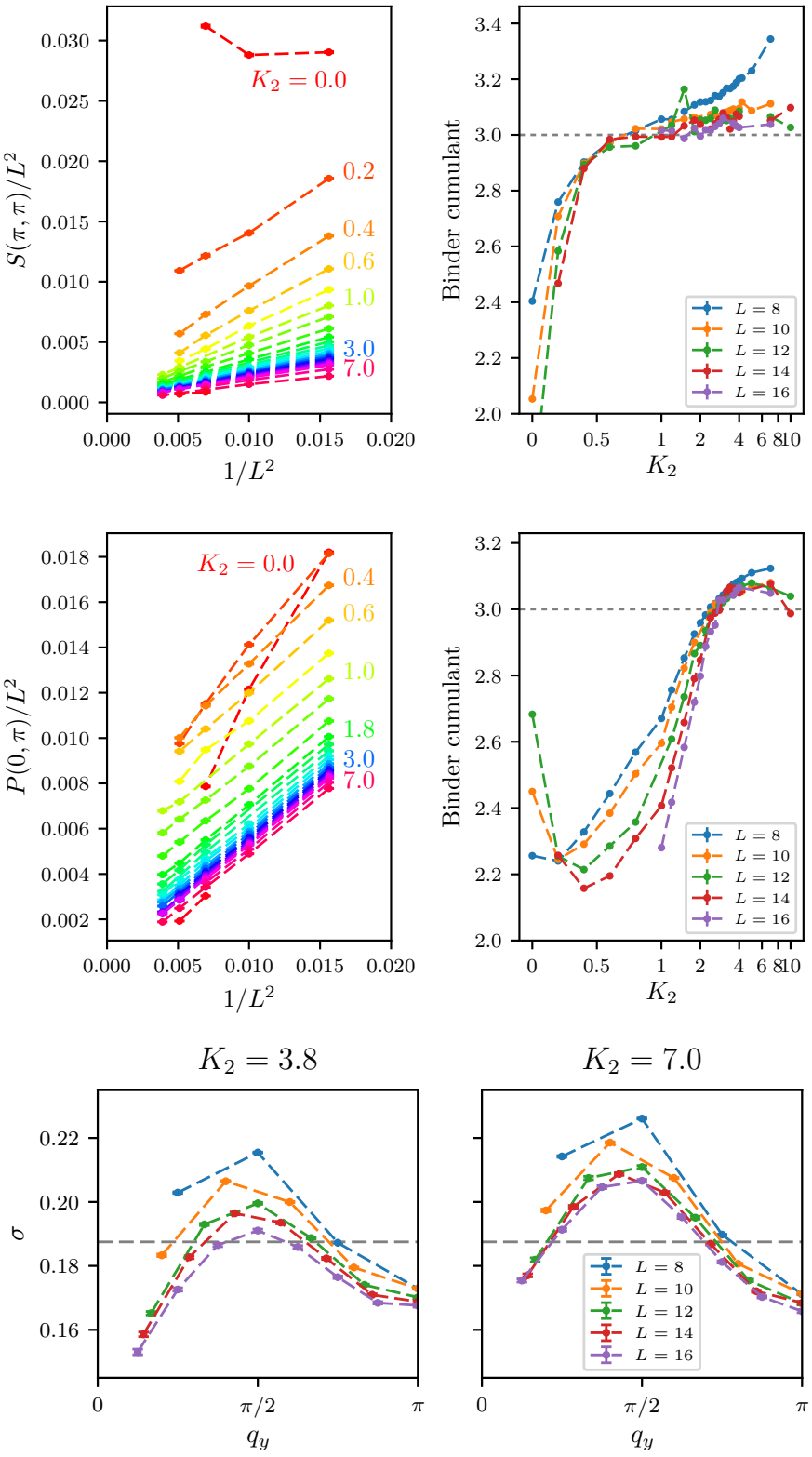


Figure 4.14: RQMC phase diagram. (Top left) $S(\pi,\pi)/L^2$ plotted against L^2 for different values of K_2. (Top right) Binder cumulant of $S(\pi,\pi)$ for system sizes from $L = 8$ to $L = 16$. Note that we use a linear scale in the range $K_2 \in [0,1]$ and a log scale for $K_2 \in [1,10]$. (Middle left) $P(0,\pi)/L^2$ plotted against L^2 for different values of K_2. (Middle right) Binder cumulant of $P(0,\pi)$ for system sizes from $L = 8$ to $L = 16$. (Bottom) Cross operator at large K_2 for different system sizes. The dashed line indicates the threshold for the EBL stability $\sigma_c = 3/16$.

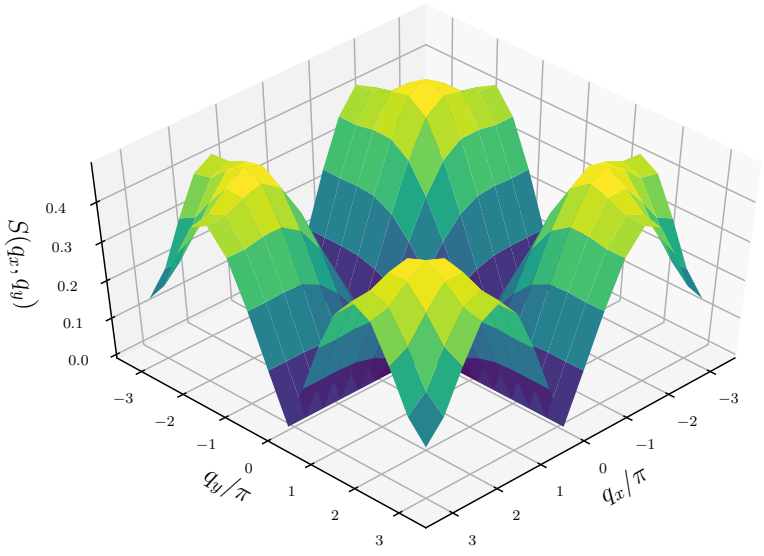


Figure 4.15: Density structure factor $S(q_x, q_y)$ obtained with RQMC simulations on $L = 16$ at $K_2 = 7$.

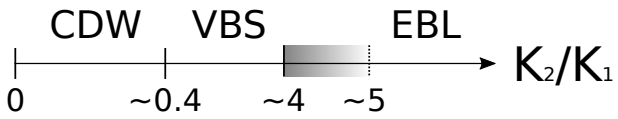


Figure 4.16: RQMC phase diagram obtained with translation-invariant RBMs with $\alpha = 1$ used as a guide from finite-size scaling on systems up to $L = 16$. The shaded region corresponds to where VBS is no longer present but where showing the stability of the EBL phase is not possible given our system size.

4.4.5 Conclusion

We established that the RBM wave functions are capable of reliably approximating the solid and liquid phases of the $K_1 - K_2$ model. In particular, increasing the number of hidden nodes (α) allows to systematically improve their precision. As compared to the variational ansatz of [Tay & Motrunich 2011] based on spin-wave theory, RBMs are not only able to be more accurate on the energy and observables even for the lowest α considered and no symmetry implemented, but they seem also capable of capturing plaquette correlations up to the thermodynamic limit, hence detecting the VBS phase. We also noticed improvements when implementing some symmetries of the model in the RBM structure. However, the finite-size trends of the order parameters calculated in VMC remain unclear in some regions which prevents the precise determination of the phase boundaries.

An exact study of the $K_1 - K_2$ model is possible thanks to reptation QMC. We showed that the presence of a guiding wave function in the form of a RBM was beneficial in many respects (shorter equilibration, faster convergence, smaller error bars). We were not able to conclusively decide on the best value of α in regard of time efficiency. Nevertheless, our finite-size scaling analysis with RQMC guided with translation-invariant $\alpha = 1$ RBMs confirmed the phase diagram found in [Tay & Motrunich 2011], thereby extending their results up to $L \leq 16$. As a point of caution, we want to stress that although the stability of EBL is visible at large K_2 and for the largest system size, we cannot exclude the possibility that for better guides or larger system sizes the criterion on the cross operator σ may not be satisfied anymore. Nevertheless, as also pointed out in [Tay & Motrunich 2011], the criterion on the cross operator σ may not be the ultimate argument to assess the stability of the EBL phase.

Regarding future developments, we are currently working on positively identifying the EBL phase from entanglement. Indeed, as showed in [Lai *et al.* 2013], the entanglement entropy is expected to scale as $\sim L \log(L)$ for bosonic systems with Bose surfaces, thus providing a possible indicator of the EBL phase. Whereas the entanglement entropy is very hard to compute in RQMC, it can in fact be accessed quite efficiently in VMC [Hastings *et al.* 2010] in particular with RBM wave functions.

Chapter 5

Reinforcement learning for quantum error correction

Contents

The quest for a viable quantum computer inevitably leads to the challenge of error correction. Indeed, qubits are fragile and can be altered by thermal fluctuations or perturbations coming from the environment. In order to make computations reliable, one has then to find ways to restore the quantum state the qubit was in before the error happened. In fact, this problem also concerns classical computers, however the peculiarities of the quantum world prevent the direct application of classical error correction schemes and therefore the field of quantum error correction (QEC) was developed.

This chapter will start with an introduction to the quantum error correction problem, with a focus on the paradigmatic toric code. Sec. 5.2 will explain how machine learning can be applied to this problem, discussing how reinforcement learning has recently become a method of choice. We will then introduce the NEAT algorithm which is an evolutionary algorithm capable of optimizing the weights and the architecture of neural-networks. Sec. 5.3 presents our preliminary results showing that the NEAT algorithm allows to find efficient error-correcting strategies for the toric code. This work has been done in collaboration with Pr. van Nieuwenburg.

5.1 Quantum error correction

Classical bits are 0s and 1s. However, the bit value stored in memory might change during computations because of thermal noise or any external perturbative process. This clearly asks for strategies to protect these bits. One possibility is to store many copies of the same bit and

a *logical bit* $0_L = 0 \overset{d}{\cdots} 0$ is then encoded by d *physical bits* (likewise we can define the logical bit 1_L). By redundantly storing the logical bit value, it is possible to easily correct errors like a *bit-flip* error on the middle bit of $0_L = 000 \rightarrow 010$: it suffices to apply the majority rule on the bit copies to retrieve the logical bit value $010 \rightarrow 000 = O_L$. Nevertheless, whenever $d/2+1$ bit-flips occur, the majority rule will return the wrong logical bit value, 1_L in that case: a *logical error* has occurred. One strategy to limit this is to increase the *code distance* d simply because logical errors happen with a probability that decreases with d.

In the quantum world, the building block of computations is the qubit which can be in a state $|0\rangle$ or $|1\rangle$ or any superposition of these two $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Groundbreaking at first sight, the new perspectives brought about by qubits are quickly limited by serious issues related to their quantumness. The first problem arises when one wants to retrieve information from a qubit $|\Psi\rangle$ since any measurement will collapse the qubit into one eigenstate of the observable, therefore corrupting the information stored in $|\Psi\rangle$. A second barrier is the no-cloning theorem which states that there is no unitary transformation that is capable of copying a quantum state. This has the immediate effect that the classical solution of redundant codes is not conceivable here. Nevertheless one possibility is to consider logical qubits encoded as $|\Psi\rangle_L = \alpha|000\rangle + \beta|111\rangle$. Another way out was found and led to the conception of so-called quantum codes like the Calderbank-Shor-Steane code [MacKay & Neal 1996] or low-density parity check (LDPC) codes [Steane 1996]. A particularly popular category of quantum codes are topological codes [Gottesman 2009] like the surface or the toric code, that is presented below.

5.1.1 Toric code

The toric code [Kitaev 2003] is a model of spin-1/2 particles living on the bonds of a square lattice with periodic boundary conditions. The Hamiltonian reads:

$$H = -\sum_v A_v - \sum_p B_p \tag{5.1}$$

which include vertex stabilizers $A_v = \Pi_{i\in v}\sigma_i^x$ where v is a vertex of the lattice (the product is done over all spins lying on nearest bonds in red in Fig. 5.1a) and plaquette stabilizers $B_p = \Pi_{i\in p}\sigma_i^z$ where p is a center of a plaquette (the product is done over all surrounding bonds in green in Fig. 5.1a). Operators A_v and B_p have ± 1 eigenvalues, commute with each other and with the Hamiltonian. As a result, the Hamiltonian is block-diagonal in the computational σ^z basis and contains $2^{2(d^2-1)}$ blocks corresponding to all the possible d^2-1 independent sets of eigenvalues ± 1[1] for the operators A_v and B_p. This means that each of these blocks is of size $2^{2d^2}/2^{2d^2-2} = 4$. In particular the ground-state is 4-fold degenerate and allows to define two logical qubits. Note that one needs $2L^2$ physical qubits to encode 2 logical qubits in the toric code which is quite a large overhead compared to other codes like LDPC codes [Steane 1996].

Ground-state manifold. The ground state of the toric code lies in the sector where $A_v = +1$ and $B_p = +1$ simultaneously for all vertices and plaquettes. The fully polarized state $|\text{FP}\rangle = |\uparrow \ldots \uparrow\rangle$ is clearly an eigenstate of the plaquette operators B_p with eigenvalue $+1$, but not an eigenstate of the vertex stabilizer A_v that flips all spins around vertex v. The symmetrized state

[1] Since we have the two relations $\prod_v A_v = 1$ and $\prod_p B_p = 1$, when PBC are imposed.

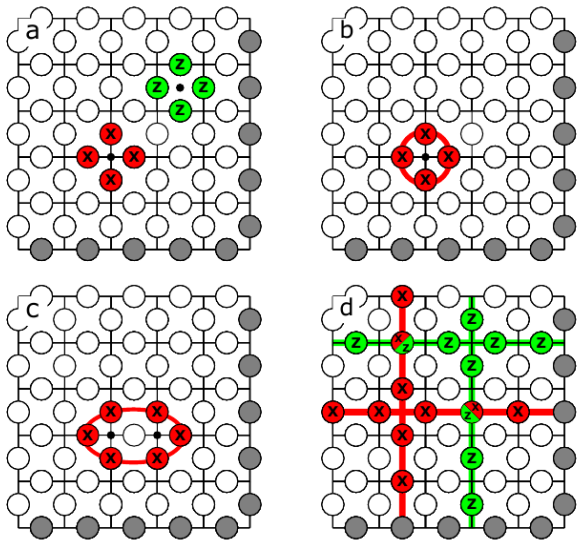


Figure 5.1: (a) Action of the two stabilizer operators A_v and B_p. (b) The action of the vertex operator can be seen as introducing a trivial loop. (c) A wider – still topologically trivial– loop. (d) The four topologically non-trivial loops of operators σ^x and σ^z allow to define the four different ground states. The figure is from [Andreasson *et al.* 2019].

$A_v|\mathrm{FP}\rangle + |\mathrm{FP}\rangle$ instead is an eigenstate of A_v with eigenvalue $+1$ while still being an eigenstate of all the B_p operators with eigenvalue $+1$. A useful representation of $A_v|\mathrm{FP}\rangle$ is to identify this state as a loop of operators σ^x (see Fig. 5.1b). Fig. 5.1c shows that the action of two A_v operators lying on two neighbouring vertices generate a bigger loop. It is not difficult to check that the state $|\mathrm{GS}\rangle = \sum_{\text{trivial loop}} \left(\prod_{i\in\text{trivial loop}} \sigma_i^x\right) |\mathrm{FP}\rangle$, i.e. that is constructed as the equal-amplitude superposition of all trivial loops acting on $|\mathrm{FP}\rangle$, lies in the ground-state manifold of the toric code. Fig. 5.1d shows that there also exist 4 types of non-trivial loops of σ^x or σ^z operators that wrap around the torus in the x or y direction. Let us denote them $X_L^{(1/2)}$, $Z_L^{(1/2)}$ according to the figure. It is possible to obtain the 3 other ground-states by applying the operators $X_L^{(1/2)}$ on $|\mathrm{GS}\rangle$, which gives the following basis set for the ground-state space: $\{|\mathrm{GS}\rangle, X_L^{(1)}|\mathrm{GS}\rangle, X_L^{(2)}|\mathrm{GS}\rangle, X_L^{(1)}X_L^{(2)}|\mathrm{GS}\rangle\}$. In fact, $X_{L,1/2}$ can be identified as the logical bit-flip operator on the logical qubit and the operators $Z_L^{(1)}$ and $Z_L^{(2)}$ allow to measure the logical state of the two qubits encoded in the toric code.

Excitations and error correction. Since the ground-state of the toric code encodes the logical qubits, it is required that the qubits stay in the ground-state manifold. However excitations may occur and move the toric code state out of the ground-state manifold. The lowest excited states are obtained by the application of a Pauli operator on $|\mathrm{GS}\rangle$. The excited state $\sigma_i^x|\mathrm{GS}\rangle$ produces a *bit-flip* error, likewise $\sigma_j^z|\mathrm{GS}\rangle$ produces a *phase-flip* error. For an excitation $\sigma^y = i\sigma^x\sigma^z$, bit-flips and phase-flips are correlated. In the following however, we will focus on the case where the noise is uncorrelated, meaning we can treat bit-flips and phase-flips independently (and completely equivalently). Also, it is common to consider an error model where errors occur with probability p on each physical qubit.

Now that errors are defined, let us see how one can correct them. In Fig. 5.2a, two σ_i^x errors produce two so-called *syndromes* that are the positions of the plaquettes having $B_p = -1$ eigenvalue (shown in orange in the figure). Contrary to Pauli operators, syndromes can be measured without altering the quantum state (since the plaquette operators B_p commute with H), thus this gives an indirect way to probe errors. One way to project the state $\sigma_i^x \sigma_j^x |\text{GS}\rangle$ back to the ground-state manifold is to apply a string of operators $\prod_k \sigma_k^x$ (blue operators in Fig. 5.2c) such that the product with the error $(\sigma_i^x \sigma_j^x \prod_k \sigma_k^x)$ forms a trivial closed loop of σ^x operators. It turns out that they are as many ways to correct an error as there are – trivial or non-trivial – loops of σ^x operators including the σ^x errors. However, Fig. 5.2d shows that a correction creating a non-trivial loop of σ^x operators actually acts as the logical bit-flip operator $X_L^{(1)}$: the corrected state is in the ground state but its logical value has changed. The central difficulty of the decoding task stems from the fact that the relation between syndromes and physical errors is not one-to-one and the inherent ambiguity of the syndromes does not allow to avoid logical errors to occur after correction.

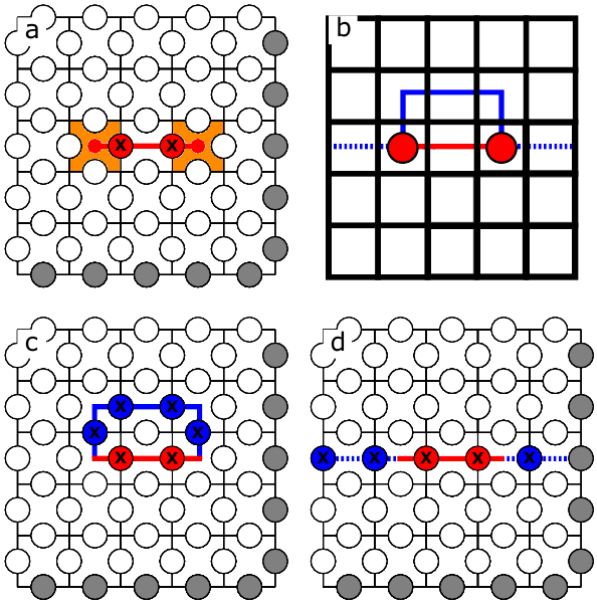


Figure 5.2: (a) Two σ^x errors occur and manifest as a syndrome represented by red dots where the plaquette stabilizers have -1 eigenvalue. (c) A possible correction is to flip spins to form a trivial loop of σ^x as shown in blue. (d) This correction creates a non-trivial loop of σ^x operators that wraps around the torus and change the logical state. The figure is from [Andreasson *et al.* 2019].

5.1.2 One example of a decoding algorithm

A perfect decoder cannot exist because of the inherent ambiguity of syndrome measurements. More problematically, the decoding problem is even proven to be NP-hard [Hsieh & LeGall 2011]. However, there exist instances of quantum codes where efficient decoding algorithms exist [Duclos-Cianci & Poulin 2010a, Poulin *et al.* 2009]. One deterministic algorithm that has near-optimal performance is minimum weight perfect matching (MWPM) [Edmonds 1965]. The decoding is achieved by linking (matching) pairs of syndromes

with strings of Pauli operators, in such a way that the total string length is minimal. This can be implemented efficiently with the Blossom algorithm [Edmonds 1965, Fowler 2015]. Fig. 5.3 shows the logical fidelity, i.e. the proportion of corrected error samples not having a logical error, as a function of physical error rate (here the probability of physical qubit-flips) for this algorithm.

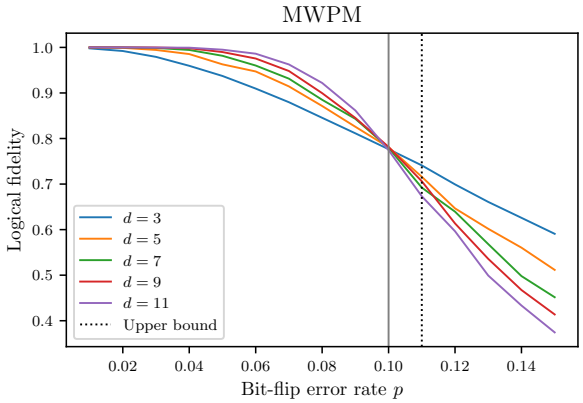


Figure 5.3: Logical fidelity as a function of bit-flip error rate for different code distances after decoding with the MWPM algorithm. The error threshold of the MWPM algorithm can be estimated around $p_c \approx 0.1$.

Fig. 5.3 shows a very interesting feature of MWPM: below an error threshold p_c the logical fidelity increases as the toric code distance increases, conversely for $p > p_c$ the trend changes and the logical fidelity worsens for larger code distances. Let us consider the limiting case $p \to 0$ to get an intuition of these trends. For p small, errors most often consist of one single qubit-flip giving rise to two syndromes on the adjacent plaquettes, the distance between these syndromes is 1 and the MWPM algorithm will never link these syndromes with a string of operators that wrap around the torus since the string length would be at least of size $d - 1 > 1$ in that case. However, when two qubit-flip errors happen say next to each other, Fig. 5.2b shows three possible correcting strings of operators: MWPM would implement the shortest (solid red) of size 2, but the correction string that wraps around the lattice (dotted blue) is only of size $d - 2 = 3$ and introduces a logical error. It is then easy to see from this example that as the code size d is smaller there will be more cases where the shortest correction induces a logical fault.

The existence of an error threshold has very important consequences on the experimental realization of the toric code since if the crafted physical qubits happen to have an experimental error rate below the MWPM error threshold, this means that it suffices to add more physical qubits (increase code distance) to obtain a higher fidelity on the logical qubit with MPWM. Note that the error threshold is a property of the decoding algorithm, therefore in general the higher the (algorithmic) error threshold the better the algorithm is considered.

Interestingly, it was shown that the toric code with uncorrelated noise can be mapped to the random bond Ising model where the error rate p plays the role of the disorder strength [Dennis *et al.* 2002]. Echoing bond percolation, the mapping reveals the existence of a critical error threshold $p_c \approx 0.11$ between an ordered phase for $p < p_c$ where the probability that a correction will create a non-trivial loop and hence alter the logical state is 0 (in the

thermodynamic limit), whereas it is 1 in the disordered phase for $p > p_c$. For finite-size codes, a crossover is visible similarly to what is shown in Fig. 5.3.

Note that there exist many other decoding algorithms, some performing better than MWPM. Maximum likely decoding (MLD) searches for the most probable error and performs better than MWPM but with a larger computational cost. Some strategies are based on tensor networks [Ferris & Poulin 2014] or on the renormalization group [Duclos-Cianci & Poulin 2010b]. Next section will introduce other decoding strategies that use machine learning techniques.

5.2 Machine learning applied to error correction

In the past years, unsupervised, supervised and reinforcement learning approaches have been applied to quantum error correction. Seminal work was done by Torlai and Melko in [Torlai & Melko 2017] where they trained a RBM to model the joint probability distribution of errors e and syndromes S as $p(e, S)$. Once the RBM is optimized, the correcting error chain e_{corr} is drawn from the marginal distribution $p(e|s)$. Other works have used feed-forward and recurrent neural networks [Krastanov & Jiang 2017, Varsamopoulos *et al.* 2020a] to learn the mapping from syndromes (input) to errors (target) in a supervised manner. Although they showed performance comparable to MWPM, sometimes better for certain error models, a major flaw of these methods comes from the possibility that the drawn correction string does not project the altered state back to the ground state manifold. In the above-mentioned works, the decoders are low-level in the sense that they tell which individual physical qubits to flip in order to correct the code. Another possibility is to consider high-level decoders, where here instead the code is projected back to the ground state with a simple deterministic algorithm and a neural-network proposes a logical correction if one is needed [Maskara *et al.* 2019, Varsamopoulos *et al.* 2020a, Chamberland & Ronagh 2018, Varsamopoulos *et al.* 2017, Baireuther *et al.* 2018]. Another series of work used reinforcement learning techniques [Sweke *et al.* 2018, Andreasson *et al.* 2019, Fitzek *et al.* 2020, Domingo Colomer *et al.* 2020], which I will explain in next section. Finally, hybrid approaches were proposed where neural-networks appear only in combination with more physically-motivated components [Liu & Poulin 2019, Ni 2020, Varsamopoulos *et al.* 2020b].

5.2.1 Decoding as a reinforcement learning problem

As explained in Sec. 2.5 of chapter 5, reinforcement learning (RL) is an optimization technique that lets an agent interact with an environment. In the QEC context, the environment is usually the state of the code and the agent can apply Pauli matrices on the physical qubits. We generally consider a decoding cycle which consists in a sequence of agent-environment interactions starting from a state containing syndromes and terminating when the agent removed all of them. Positive rewards are given if the correction did not introduce a logical error, negative otherwise.

Note that contrary to supervised learning, RL does not provide examples of solved cases which potentially enables the discovery of innovative decoding strategies. RL seems also more naturally suited to decoding tasks because of its sequential nature, that resembles what an actual correction process could look like[2]. We also expect RL to find decoders that will be more interpretable than the ones obtained in supervised learning: the correcting strategies can be

[2]RNN can in fact also be used within the supervised approaches [Baireuther *et al.* 2018, Varsamopoulos *et al.* 2020a].

accessed more easily by evaluating which action is preferable in which scenario. Setting up the problem in a sequential way could also improve the performance since the agent is in principle able to correct the effect of bad actions (it may have taken previously) along the decoding cycle. More importantly, it enforces time-translation invariance which allows a lot of information reuse: starting from an initially complex syndrome pattern, the complexity of the syndrome pattern is expected to decrease with time and the decoding agent can then use the knowledge it acquired independently by decoding simpler initial syndrome patterns.

In general, an agent chooses actions based on a policy π, which is in essence its decoding strategy. More precisely, given a state of the environment S the agent chooses action a with probability $\pi(a|S)$. Optimization of π can be achieved within the framework of so-called Q-learning [Sutton & Barto 2018] which was used in all prior works [Sweke *et al.* 2018, Andreasson *et al.* 2019, Fitzek *et al.* 2020, Domingo Colomer *et al.* 2020]. We take a different path in our work as we will employ evolutionary algorithms to perform the optimization directly in policy space. In our approach, the agent's policy π is approximated by a neural network f, which returns the selected action a given the current code state S, i.e. $f(S) = a$. The neural network optimization is then done with the NEAT algorithm explained below.

5.2.2 The NEAT algorithm

The NeuroEvolution Augmented Topologies (NEAT) algorithm was introduced by Stanley and Miikulainen in [Stanley & Miikkulainen 2002] and is part of the family of evolutionary algorithms. These algorithms are gradient-free optimization techniques that work by evolving a *population* of individuals according to heuristics inspired by biological evolution. At each generation – optimization step – the individual's fitness is evaluated and the population of the next generation is obtained by *mutation*, *reproduction* and *selection* of the best performing individuals. Mutation consists in randomly changing some of the properties of an individual. Selection is based on the individual's *fitness* and only the best performing elements of population will survive to the next generation. Reproduction will allow mixing of two individuals to create an offspring that inherits some of their respective characteristics. Thanks to these heuristics, the individuals are more and more fitted to their environment as evolution goes on.

In the NEAT algorithm, the evolution is done on a population of neural networks. The specificity of NEAT is that mutations not only change the internal weights of the neural networks but also their own architecture. Indeed, nodes can be added as well as connections between nodes. The breakthrough of the NEAT algorithm was to make possible a meaningful breeding of neural networks of different topologies (architectures). NEAT works as a combination of the three following ideas: (i) the existence of *historical markings* in the genetic encoding of neural networks allows meaningful crossover, (ii) isolating subgroups –niches– of similar individuals in the population allows to protect innovation, (iii) starting evolution with simple neural networks allows to obtain a final solution of minimal complexity. NEAT showed very good performance on different control tasks benchmarks and demonstrated very fast convergence to a neural network solution of small complexity compared to previous works at the time. This approach has been the basis of many important and recent contributions in the field of neuroevolution [Real *et al.* 2017, Real *et al.* 2019, Liu *et al.* 2018].

In the following paragraphs, we tried to make the description of the algorithm as self-contained as possible, nevertheless we have eluded some technical details that can be found

in [Stanley & Miikkulainen 2002].

Genetic encoding and crossovers. Each neural network of the population is encoded by a genome as shown in Fig. 5.4a. The key insight of [Stanley & Miikkulainen 2002] was to introduce an innovation number that keeps track of the history of a gene. Every new connection appearing in the population (see Fig. 5.4b) via a mutation is assigned a unique identification number (note that weight mutation does not generate a new innovation number). This crucially enables a simple and meaningful procedure for the crossover of two neural-networks as shown in Fig. 5.4c.

Protection of the innovation by speciation. Another key element of NEAT is the design of a speciation mechanism that allows subgroups of similar neural networks – species – to evolve separately from the rest of the population. When the architecture of a neural network is changed via a mutation, it is likely that it will not perform well at first and a few generations are needed so that its weights can be adjusted. The issue is that the selection rules will eliminate these more complex individuals and effectively prevent better topologies to be found. It is possible to circumvent this issue by creating niches of individuals that share characteristics among themselves but not with the rest of the population and applying selection independently on these subgroups. As a result, speciation is able to protect genetic innovation.

In [Stanley & Miikkulainen 2002], the species are defined via a compatibility distance δ which simply accounts for the number of excess E or disjoint D genes between two genomes, as well as the average weight differences in the matching genes $\overline{W}$:

$$\delta = c_1 \frac{E}{N_{\text{genes}}} + c_2 \frac{D}{N_{\text{genes}}} + c_3 \overline{W} \tag{5.2}$$

where c_i are hyperparameters to adjust the respective importance of these three factors, N_{genes} is the number of genes in the largest genome. At each generation, genomes are sequentially placed in species by checking whether the compatibility δ between the current genome and a genome randomly picked from a given species is below a threshold distance δ_c. Additionally, NEAT employs a heuristic called explicit fitness sharing which favors homogeneity inside the species. The idea is to fight against the tendency that largely-populated species take over the rest of the species. This works by adjusting the size N_j of species j according to the ratio:

$$N'_j = N_j \frac{\overline{f_j}}{\overline{f}} \tag{5.3}$$

where N'_j is the size of species j for the next generation, $\overline{f}$ is the fitness averaged over the entire population and $\overline{f_j}$ averaged over the individuals in species j.

Minimizing dimensionality. The last key insight of [Stanley & Miikkulainen 2002] was to initialize the population with neural networks having the simplest topology possible. For instance, neural networks of the first generation may have no hidden nodes. In combination with speciation, this is argued to minimize the complexity of the final solution. Indeed, new architectural components are tested and optimized independently thanks to speciation: if the architectural innovation is proven to provide a significant performance boost, it is then included in the rest of the population. This way the complexity of the population only increases when

necessary. Starting the evolution with the simplest neural networks then ensures that the final solution has the minimal complexity to achieve the given task.

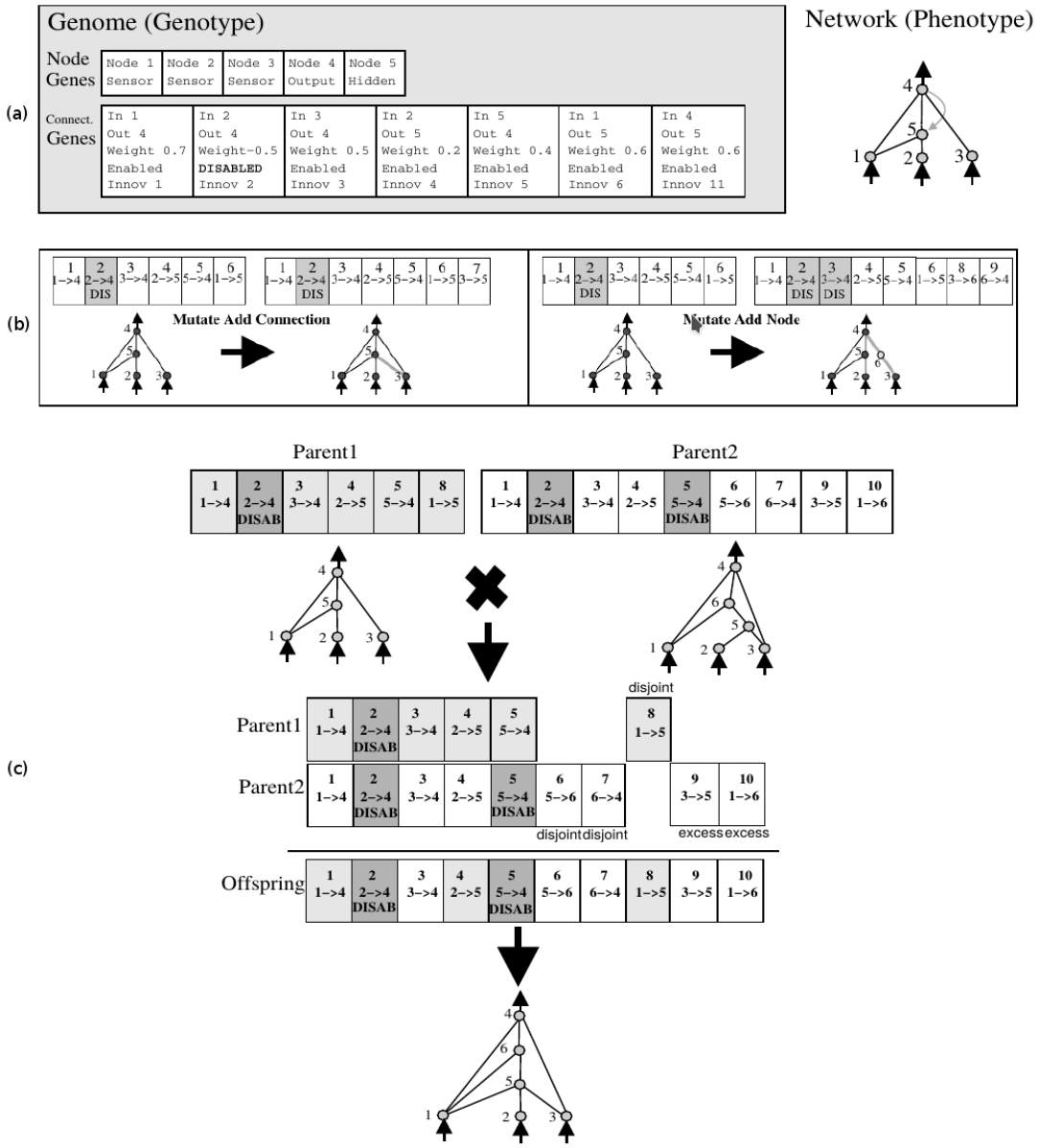


Figure 5.4: (a) The genome of a neural network contains node and connection genes. A node gene stores an identification number and its type (input (sensor), hidden or output). A connection gene informs about which nodes it connects (the directionality allows to define recurrent connection that creates a loop in the neural network structure), the weight value it carries, a boolean allowing for disabling the connection and, crucially, the *innovation number* (see main text). All these information uniquely define a phenotype neural network. (b) (Left) Adding a connection is done by adding a new connection gene to the genome, with a new unique innovation number (here 7). (Right) Adding a node is done by splitting an existing connection in two where the previous connection (here $3 \rightarrow 4$) is disabled and two new connection genes are created (here 8 and 9). (c) Crossover is achieved by matching connection genes that share innovation numbers between the two parents (here genes 1-5), these matching genes are transmitted to the offspring with a weight and disabling option that is picked with equal probability from one of the two parents. The other disjoint genes are inherited randomly by the offspring. The figures are from [Stanley & Miikkulainen 2002].

5.3 A NEAT quantum error decoder

The properties of the NEAT algorithm are very appealing in the QEC context. First, the fact that it can optimize neural-network architectures makes it very flexible: the pecularities of quantum devices may likely change drastically in the near future, so it could potentially save a lot human effort to leave the task of finding the appropriate neural-network structure to the NEAT algorithm. Second, by keeping the architecture as small as possible (only necessary components survive evolution), NEAT provides fast-to-execute decoders, which can be crucial in real systems where the proliferation of errors might simply overcome too slow decoding algorithms. Third, shallow neural networks are also more easily interpretable which could possibly give insights on new correcting strategies. Finally, training is expected to be fast because the method is gradient-free and highly parallelizable (since independent individuals in the population are evaluated on independent syndromes). Our work focused on the ability of NEAT to decode the toric code with uncorrelated noise. After presenting the training setup in details, we will highlight some of the preliminary results we obtained in the next sections.

5.3.1 Training setup

Input data. Since we consider only σ^x bit-flip errors, the input data include only the L^2 eigenvalues of the plaquette operators B_p, expressed as binary numbers $B_p \in \{0, 1\}$. At first, the input data contained information about which physical qubits were already flipped (adding $2L^2$ more binary values), thus providing the history of the past actions to the agent. Surprisingly enough, we were able to achieve the same level of performance without memory of the past. This somewhat echoes the ability of the NEAT algorithm to achieve the difficult task of double pole balancing without information on the velocities of the poles [Stanley & Miikkulainen 2002].

Exploiting translation-invariance. Translational-invariance of the toric code can be used to simplify the decoding task: one can use CNNs as in [Ni 2020] or implement *perspectives*, introduced in [Andreasson *et al.* 2019] (also used in [Fitzek *et al.* 2020, Domingo Colomer *et al.* 2020]). As shown in Fig. 5.5, for a sample having N_S syndromes, the perspectives are the N_S translated copies of the sample such that for each of these copies there is a syndrome residing in the central plaquette of the lattice. As a result, the neural networks will always be fed with syndrome patterns containing a syndrome in the central plaquette. The advantage of this formatting is that the action space can be reduced to a constant size of 4, corresponding to applying a σ^x operator on the 4 physical qubits the nearest to the central plaquette. Thus, the correction will always proceed by flipping qubits that are next to syndromes. Although this introduces a bias, we think this assumption is reasonable and provides significant speed-up since there is at least one flipped qubit out of the four surrounding a plaquette containing a syndrome. Note that we could further reduce the action space (down to 1) by using similarly the rotation or reflection symmetries of the toric code, however it is important to keep in mind that real experiments are usually not perfectly symmetric. We could in principle keep the perspective tricks and take into account imperfect translation symmetries by considering faulty syndrome measurements for example.

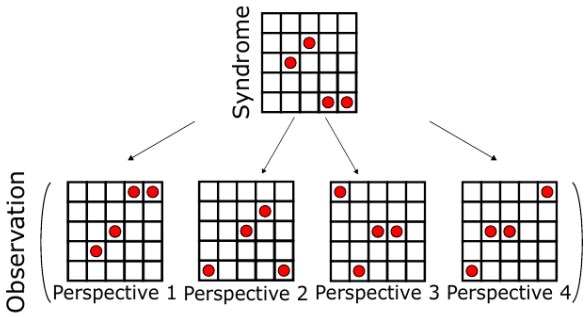


Figure 5.5: Each error sample generates as many translated "perspectives" as syndromes, where each perspective has a central syndrome. Figure taken from [Andreasson *et al.* 2019].

Decoding algorithm. Starting from an initial observation i.e. measurements of the plaquette stabilizers $B = (B_1, \ldots, B_N)$, a decoding cycle is performed as follows:

Data: Toric code Code, Neural network f
Result: reward$\in \{0, 1\}$
while *Code has syndromes* **do**
  Measure plaquette stabilizers: **B**$= (B_1, \ldots, B_N)$
  for *i such that* $B_i = -1$ **do**
    Generate the ith perspective P_i
    for *action a in* ***A****=(flip1,...,flip4)* **do**
      Compute the action probability p_a from the policy network f: $Prob_a$=f(P_i)
    end
  end
  if *rand()* $< \varepsilon$ **then**
    action = random action in **A**
  else
    action = $\underset{a}{\text{argmax}}$ Prob$_a$
  end
  if *action was already taken* **then**
    return reward=0
  else
    Code is changed according to action
  end
end
if *Code has a logical error* **then**
  return reward=0
else
  return reward=+1
end

Note that we also prevented the decoder from performing an action twice on the same qubit (thus a decoding cycle cannot last more than $2L^2$ steps). An ε-greedy policy is used and consists in picking a random action with probability ε instead of the action recommended

by the neural-network policy with highest probability, we found it worked best with $\varepsilon = 0.1$. Notice that our approach is not biased towards selecting the smallest error-correcting chain like MWPM or as in [Andreasson *et al.* 2019, Fitzek *et al.* 2020], neither towards selecting the most probable error patterns like MLD or as in [Torlai & Melko 2017], since the reward is only a function of the presence or absence of a logical error in the final state (similarly to [Domingo Colomer *et al.* 2020]). Note that checking whether a logical error occurred or not can be done quickly by measuring the logical operators $Z_L^{(1)}$ and $Z_L^{(2)}$ introduced in Sec. 5.1.1.

Fitness evaluation. The fitness of each neural network is evaluated by presenting N^{eval} error samples, so-called *puzzles*. Each puzzle is generated by inserting a σ^x bit-flip with probability p on each site, hence the total number of bit-flips follows a binomial distribution with parameters $n = 2L^2$ (number of qubits) and p. As done in previous works, the training set is made as diverse as possible to ensure good performance across different error rate regimes. In our case, the N^{eval} training samples are obtained at error rate $p \in \{0.01, 0.05, 0.1, 0.15\}$, i.e. in practice $N^{\text{eval}=0.01} = N^{\text{eval}=0.05} = N^{\text{eval}=0.1} = N^{\text{eval}=0.15} = N^{\text{eval}}/4$. This way, the neural network fitness reflects its ability to perform in situations of varying difficulty. However, as training lasts, the population of neural networks gets better on average and the easiest puzzles are solved by a majority of them. Spending time on these easy cases is somewhat useless, this suggests that we could instead adapt the set of problems presented to the neural networks and purposefully include only puzzles in the training set the neural networks struggle more on. This strategy is called *curriculum learning*, however we found that it only improved marginally the performance while slowing down the training process. In all simulations, N^{eval} was set to 400 training puzzles.

Population characteristics. The first generation is composed of N_{pop} (100 in most simulations) fully-connected neural networks with no hidden nodes, the $4L^2$ weights are initialized at random (from a Gaussian noise). The mutation rates are set to 0.1 for the addition of a connection or a node, to 0.5 for the weights, which we obtained from a hyperparameter grid-search.

5.3.2 Error correction performance

A common way to measure the performance of a decoder is to track the logical fidelity as a function of physical error rate. This quantity is computed as the ratio of successfully decoded samples (reward 1 returned from the algorithm above) over the total number of samples. Fig. 5.6 shows the performance of the best neural-network found by the NEAT algorithm after a few hundreds of generations. Note that the NEAT evolutions are run separately for different code distances, which in principle makes the definition of an algorithmic error threshold ill-posed since each run is not constrained to converge to the same decoding algorithm.

The decoders found by NEAT display features shared by MWPM and most of decoders: the performance deteriorates as the physical error rate is increased and a crossing of the curves is visible around $p_c \approx 0.08 - 0.09$ for NEAT, which is a little worse than MWPM that has $p_c \approx 0.1$, but the critical fidelity ≈ 0.85 is better than ≈ 0.75 for MWPM. Moreover, one can see that the logical fidelity is greater than MWPM for the largest error rates beyond $p = 0.1$. Despite the simplicity of our approach, we are able to reach the same performance level than previously [Torlai & Melko 2017, Andreasson *et al.* 2019] (same code and same type of noise), outperforming MWPM on bit-flip errors in the noisiest regime. These results are obtained with

considerably less computational power than previous works as well as returning the most succint neural-network decoders. Indeed, as can be seen in Tab. 5.1, our policy neural networks have 1000 to 10000 times fewer parameters than deep Q-networks trained in previous works using Q-learning[3]. Fig. 5.7 shows the policy-network obtained for $d = 3$ which contains 8 hidden nodes and a total of 47 weights.

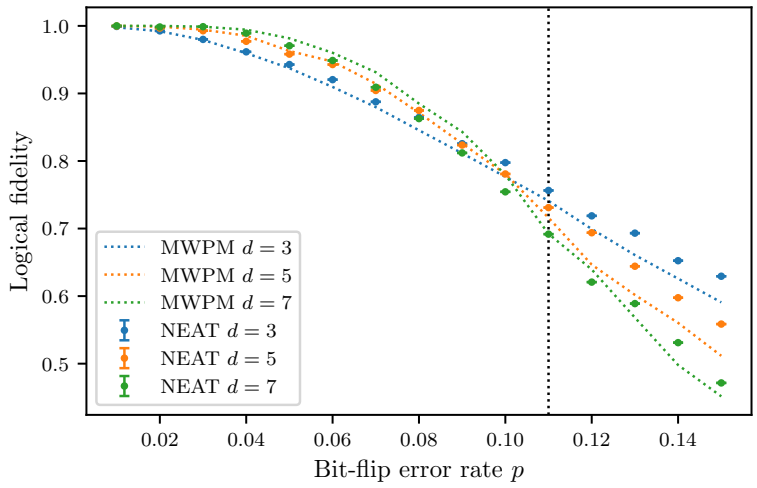


Figure 5.6: Logical fidelity as a function of physical error rate p for different code distances. The performance of MWPM is shown in dotted lines and symbols show the performance of the best neural-network found by the NEAT algorithm. The vertical dotted line shows the theoretical upper bound of the error threshold [Dennis *et al.* 2002]. Evaluation is done on 5000 independent random error samples for each physical error rate.

	$d = 3$	$d = 5$	$d = 7$
[Sweke *et al.* 2018]	$\sim$ 800000	$\sim$ 2000000	$\sim$ 3800000
[Andreasson *et al.* 2019]		$\sim$ 500000	$\sim$ 1200000
[Fitzek *et al.* 2020]		$\sim$ 900000	$\sim$ 9000000
[Domingo Colomer *et al.* 2020]	$\sim$ 640000	$\sim$ 1700000	$\sim$ 3200000
Our work	$\sim$**110**	$\sim$**300**	$\sim$**550**

Table 5.1: Number of parameters of the deep Q-networks and of the policy-neural-networks found by the NEAT algorithm.

[3]It should be noted though that [Fitzek *et al.* 2020, Domingo Colomer *et al.* 2020] treat depolarizing noise and [Sweke *et al.* 2018] deals with faulty measurements, which are harder decoding tasks.

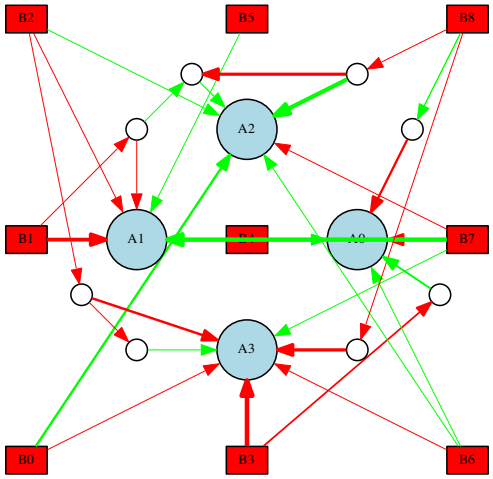


Figure 5.7: Architecture of a $d = 3$ NEAT decoder. Plaquette operators (in red boxes) are placed where they are located on the lattice. The width of the edges is proportional to the corresponding absolute value of the weight. Positive (negative) weighting is shown in green (red). The 4 output nodes (blue filled circles) are placed on the location of the qubit they flip. The hidden nodes are displayed as white circles.

5.3.3 Transfer learning by genomic transplantation

Thanks to the use of perspectives, it is actually possible to use a decoder trained on a small code to treat larger codes. This can be done by performing what we call *genome transplantation*. Transplantation simply consists in creating a neural network NN_2 applicable to code distance d_2 from a neural network NN_1 that was trained on code distance $d_1 < d_2$. This works by adding $d_2^2 - d_1^2$ input neurons to NN_1 that corresponds to plaquette operators the furthest away from the central plaquette, which do not exist in the d_1-code. All the weight connections coming from these neurons are set to 0, therefore the region beyond a distance of $\frac{d_1}{2}$ lattice spacing is effectively ignored during decoding. The resulting transplanted neural network is showed in Fig. 5.8(a) with $d_2 = 5$ and $d_1 = 3$ with NN_1 being the neural network showed in Fig. 5.7. Fig. 5.8(b) shows the performance of such transplanted genomes starting from a neural network trained at $d = 3$.

In the limit of small error rates, it is expected that the *transplanted* decoders perform well since in that regime, there are only a few qubit flips, each separated by a distance that grows on average with code distance. Therefore the fact that the transplanted neural networks ignore long-distance information has little effect in that regime. The transplantation performs quite well until the logical fidelity slowly drops with increasing error rate. It is quite remarkable that having only one NEAT decoder trained at $d = 3$ is sufficient to define decoders for arbitrary large code distances with zero additional training cost and performance that is not null although far from optimal.

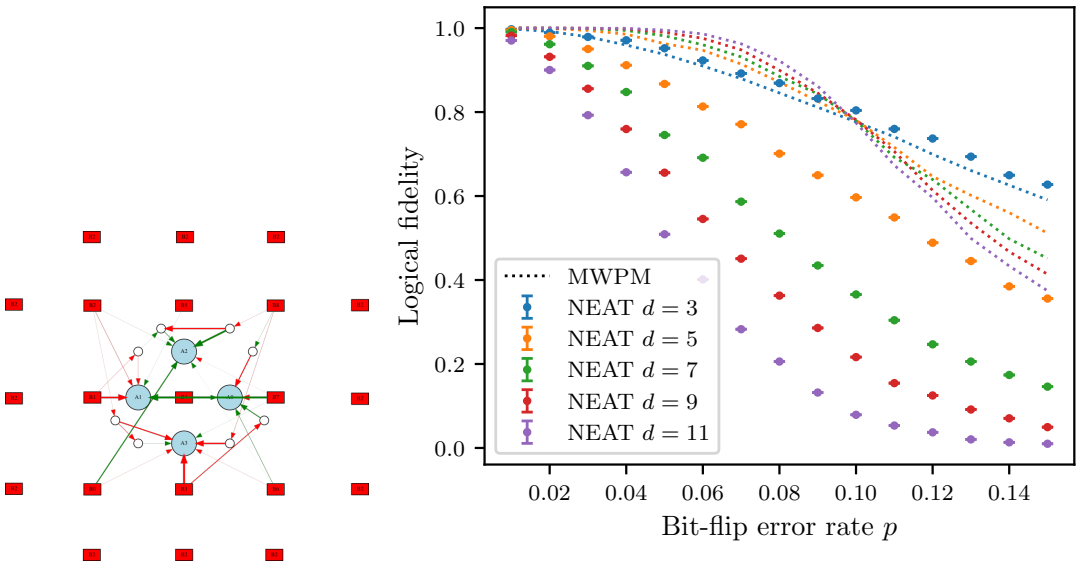


Figure 5.8: (Left) A neural network obtained from training at $d = 3$ can be used as a $d = 5$ decoder by inserting plaquette input nodes without connection weights linked to the rest of the neural network. (Right) Logical error rate as a function of bit-flip noise for different code distances from a NEAT decoder trained at $d = 3$ and its transplanted counterpart for the evaluation on larger code distances. The dashed lines indicate the ratio of error configurations that contain no bit-flips. The evaluation is done over 5000 random syndromes for each noise rate.

Speeding up trainings. Another direct application of genomic transplantation is the possibility to initialize a population of $d = 5$ decoders with the best $d = 3$ decoder found in a previous NEAT run. Fig. 5.9 shows that it can accelerate a lot training, in particular for the largest system sizes.

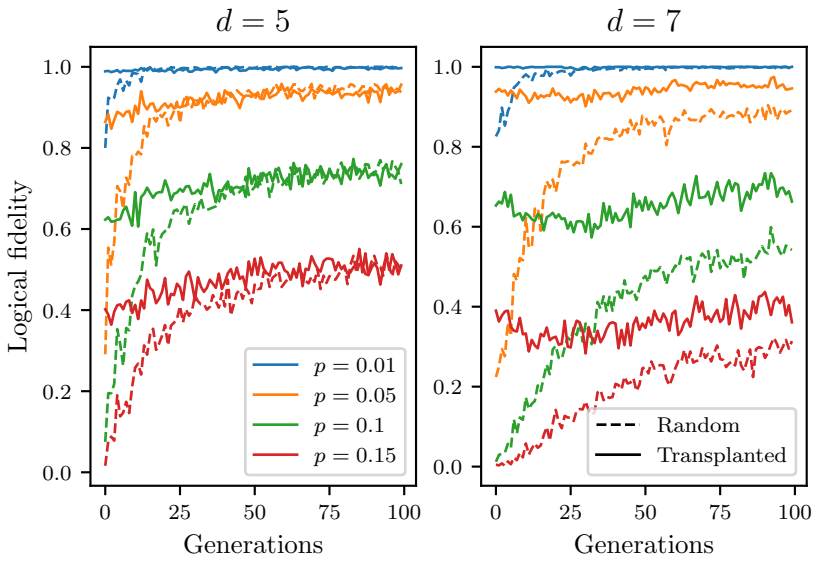


Figure 5.9: Logical fidelity against time (number of generations) for the best individual of each generation evaluated on 1000 random puzzles at physical error rates $p = 0.01, 0.05, 0.1, 0.15$. The dashed lines correspond to starting evolution with an initial random population, while solid lines correspond to starting with a population of *transplanted* neural-networks from the best $d_1 = 3$ decoder for $d_2 = 5$, and the best $d_1 = 5$ decoder for $d_2 = 7$.

5.3.4 Future work

In summary, we showed that the NEAT algorithm can reproduce the same level of decoding performance than MWPM or approaches based on Q-learning. We are confident that we can extend these preliminary results to larger system sizes, in particular this is made possible thanks to genomic transplantation (see last section) that allows to start the evolution with a good starting population. Crucially, by performing optimization directly in policy space and thanks to the properties of the NEAT algorithm, we were able to achieve the decoding task with very small neural networks, which represents a gain of the order of 10^4 in terms of number of network parameters. This not only provides a fast decoder but makes the interpretation of the policy possible. We plan to study in more depth the policy-networks found by NEAT (such as the one shown in Fig. 5.7) and attempt to understand their decoding strategy.

Regarding future developments, it would be interesting to see how these performances translate to harder QEC scenarios with depolarizing noise or fault-tolerant computations for instance. We could also test recurrent neural networks on these tasks given they can reuse information from the past in a more natural way than in our setup. The only element to change would be to enable the creation of loops in the neural networks evolved by NEAT. Performance improvements could also be expected from the use of policy gradient methods [Williams 1992]. The idea would be to first perform a NEAT run, keep the best policy-network and apply a few policy gradient steps to optimize it further. Alternatively, we could also examine whether the hyper-NEAT algorithm [Stanley *et al.* 2009] could improve on the situation as it allows in principle to discover and exploit the symmetries of the task in an automatic manner.

Chapter 6

Conclusion

Searching for new efficient classical algorithms in the era of quantum computers may seem anachronistic at first glance. Unfortunately (or fortunately), despite recent progress[Arute *et al.* 2019], quantum supremacy is far from being established[Preskill 2018]. At the same time, in the last decades, the field of machine learning has developed a set of extremely efficient tools (algorithms, softwares and even hardwares) for the treatment of high-dimensional functions. The biggest success came from deep learning and over the years neural-networks have proved to be one of the most flexible and efficient function approximators.

In short, my work consisted in evaluating the opportunity of using neural networks in three problems of condensed matter physics: (i) the classification of quantum phases of matter, (ii) the variational representation of quantum states, and (iii) the correction of errors in quantum codes.

Neural-networks as phase classifiers

Formulating the problem of detecting a phase transition as a classification task was a major paradigm shift in the field of condensed matter theory [Carrasquilla & Melko 2017]. Although exciting prospects were highlighted such as the discovery of new unknown order parameters, our work [Théveniaut & Alet 2019] raises concerns about the reliability of this approach.

Our results evidence that the ML predictions inevitably suffer from the same pitfalls as any application of machine learning. We found that the specific hyperparameter settings (choice of activation functions, addition of dropout or regularization) have sometimes a great impact on the final outcome. Moreover, for all setups considered, we observed dramatic overfitting to the training data: the neural-networks capture features specific to the system size or disorder strength of the training samples. This is surely a major issue in our approach since the goal is to eventually predict the behaviour of the system near the transition in the thermodynamic limit. Despite our best efforts to be as agnostic and general as possible, we are also not sure whether the advantages of our setups (possible interpretation, consistent finite-size trends) would hold for an alternative input formatting and/or a different neural network architecture that would be equally well justified.

One could reply to the previous remarks that searching for better hyper-parameters or adopting the unsupervised [Wang 2016, Wetzel 2017] or semi-supervised [van Nieuwenburg *et al.* 2017, Broecker *et al.* 2017a] methods may improve on the situation. The reason we think this is not true stems from the more general interrogation: how much can we trust the predictions of a neural network? Over the years, there has been an accumulation of evidences showing that neural-networks are not only notorious black-boxes but are also fragile [Goodfellow *et al.* 2015] and their generalization power can be seriously questioned [Zhang *et al.* 2017]. On the other hand, ML can be viewed as a functional

approximation technique whose output is conditioned by an objective function, the class of functions considered and so on, which makes it inherently *biased*. Retrospectively, it seemed very ambitious to combine such an approximative and general approach with the precise and physically-motivated finite-size scaling method. All the more so as we studied a transition whose scaling theory is still not established and controversial [Laflorencie *et al.* 2020].

We see different outlooks regarding the future of machine learning applied to the study of phases of matter and phase transitions. First, progress may be tied to improvements coming from the ML community. In particular a lot of effort has been directed lately towards the design of efficient interpretation methods, stimulated by the ethical and legal issues raised by the opacity of neural-networks in industrial applications. We think interpretability of the predictions will set the next milestone for automatic phase classification and we can already see some progress in this direction [Liu *et al.* 2019, Greitemann *et al.* 2019, Dawid *et al.* 2020, Cole *et al.* 2020]. Finally, the current situation may perhaps benefit from the interesting connections uncovered between renormalization group theory and machine learning [Mehta & Schwab 2014, Koch-Janusz & Ringel 2018, Li & Wang 2018, Iso *et al.* 2018, Lenggenhager *et al.* 2020].

Neural-networks as quantum wave functions

The seminal idea of [Carleo & Troyer 2017] was to connect neural-networks and their impressive capacity to compress the information contained in large datasets, to the central problem of quantum many-body physics which is the exponential complexity of quantum states. The parametrization of quantum wave functions with neural-networks showed promise early on, with great empirical success [Carleo & Troyer 2017, Nomura *et al.* 2017] and theoretical demonstrations of the efficiency and expressive power of these representations [Gao & Duan 2017, Deng *et al.* 2017b]. Following up on this, we used RBM states as variational ansätze and guiding wave functions for a VMC and reptation QMC study of a 2d constrained model of hardcore bosons introduced in [Tay & Motrunich 2011].

We found that RBM states are expressive enough to capture the two solid and the liquid phases in this model and achieve considerably higher precision than the physically-motived ansatz considered in [Tay & Motrunich 2010]. Our exact study based on reptation QMC confirms the phase diagram of [Tay & Motrunich 2011] for system sizes up to $L = 16$. Used as guiding wave functions, RBM wave functions allow for drastic improvements (shorter equilibration and faster convergence). Nevertheless, we were not able to conclude on the existence of the EBL phase at large K_2 based on the stability criterion proposed in [Tay & Motrunich 2011]. This criterion may in fact be questioned as the authors of [Tay & Motrunich 2011] admit and therefore a definitive conclusion on the EBL requires a better probe. We think considering entanglement properties may be a step forward, indeed detecting a $\sim L \log L$ scaling of the entanglement entropy at large K_2 could provide a direct and positive proof of the existence of an EBL phase.

In the near future, NQS may well benefit from further transfer of knowledge from the ML community (optimized softwares, GPU parallelism for example) and we can already see good effort in this direction [Carleo *et al.* 2019a]. Another direction of research concerns the challenging task of implementing and efficiently simulating fermionic symmetries in NQS. Finally, understanding the effect of the neural-network architecture on the approximation or describing the set of all physical states NQS can encode efficiently are still unsettled issues. A solution

would provide a firm theoretical basis for NQS which could then stand among the most flexible and efficient variational ansätze.

Neural-networks as quantum error decoders

The use of reinforcement learning techniques in the field of quantum error correction was inevitable. On the one hand, QEC can be naturally formulated as a two-player game with incomplete information: first, one player flips qubits, then the other player receives the measured syndromes and attempts correction, the game is won if the original logical state is preserved. On the other hand, RL has recently showed impressive performance in game-playing tasks [Silver *et al.* 2016, Silver *et al.* 2017, Silver *et al.* 2018]. In a series of recent works, the match indeed proved successful [Sweke *et al.* 2018, Andreasson *et al.* 2019, Fitzek *et al.* 2020, Domingo Colomer *et al.* 2020]. We propose a follow-up study based on sligthly different techniques as we used evolutionary algorithms, in particular the NEAT algorithm [Stanley & Miikkulainen 2002].

Our results showcase the capability of our approach on the toric code with uncorrelated noise, as we are able to reach performance levels equivalent to MWPM or other ML-based techniques [Torlai & Melko 2017, Andreasson *et al.* 2019]. More strikingly, this is achieved with neural-networks containing a number of parameters that is 4 orders of magnitude smaller compared to the deep networks considered previously. This clearly shows that *depth is not needed for this task*. Several consequences follow: (i) decoders are fast to execute which could prove crucial in real-life applications, (ii) we expect a more favorable computational scaling to access larger systems and (iii) it might possible to understand the neural-network decoding strategies.

However, we want to point out that the algorithmic complexity of our algorithm is at least $O(L^2)$ (corresponding to evaluating the output of a neural-network), which does not at all compete with the best known deterministic algorithm of complexity $O(\log L)$ [Duclos-Cianci & Poulin 2010a]. Nevertheless, we think a machine-learning treatment for QEC could well find its place in situations where efficient hand-made algorithms do not exist. In particular, the ML flexibility is a definitive practical advantage as the specific experimental realization of quantum codes might likely change a lot in the future. Finally, our work provides a proof of concept in line with the recent fruitful combination of evolutionary strategies with traditional machine learning [Salimans *et al.* 2017, Stanley *et al.* 2019]. We hope this can stimulate further contributions in these promising directions.

Perspectives

Let us conclude with broader perspectives on the interplay between ML and quantum physics. We think that an important research direction will be to couple ML techniques to experiments, such as in the field of material design [Liu *et al.* 2017b, Schmidt *et al.* 2019] or quantum control [Bukov *et al.* 2018, Niu *et al.* 2019]. The interaction between quantum physics and machine learning will also certainly take place through theoretical contributions, with the emergence of new research areas such as quantum machine learning [Schuld *et al.* 2015, Biamonte *et al.* 2017] or the use of statistical physics for the understanding of neural networks [Spigler *et al.* 2019, Geiger *et al.* 2020]. We hope that our work contributes to the first steps of this cross-fertilization, showing at the same time the interest and current limitations of machine learning for studying complex quantum systems.

Appendices

Appendix A

Complexity of the algorithms

Contents

The complexity of the VMC and RQMC methods are evaluated in this appendix. We will examine the computational cost associated to the three types of RBMs we considered in chapter 3: rRBMs with $M \equiv \alpha N$, $\mathcal{T}$-invariant rRBMs with $S = N$ and $M \equiv \alpha$ and C_{4v}-invariant rRBMs with $S = 8N$ and $M \equiv \alpha$. The complexity is studied as a function of the system size N and the number of variational parameters with α .

Evaluating $\Psi(\sigma_1^z, \ldots, \sigma_N^z)$. Let us consider particle-hole symmetric RBM ansatz without biases ($a_i = 0$, $b_j = 0$) and recall the expression of the non-symmetric RBM ansatz with M hidden nodes:

$$\Psi(\sigma_1^z, \ldots, \sigma_N^z) = \prod_{j=1}^{M} \cosh\left(\sum_{i=1}^{N} W_{ij}\sigma_i^z\right) \tag{A.1}$$

The symmetric ansatz with S symmetries and M filters writes:

$$\Psi(\sigma_1^z, \ldots, \sigma_N^z) = \prod_{s=1}^{S}\prod_{f=1}^{M} \cosh\left(\sum_{i=1}^{N} W_i^{(f)}\widetilde{\sigma_i^z}(s)\right) \tag{A.2}$$

Importantly, let us denote

$$\theta_j(\boldsymbol{\sigma}^z) = \sum_{i=1}^{N} W_{ij}\sigma_i^z \tag{A.3}$$

$$\theta_{f,s}(\boldsymbol{\sigma}^z) = \sum_{i=1}^{N} W_i^{(f)}\widetilde{\sigma_i^z}(s) \tag{A.4}$$

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Calculating $\widetilde{\sigma_i^z}(s)$ for all i and s	-	$\propto N^2$	$\propto 8N^2$
Calculating $\theta_j(\boldsymbol{\sigma}^z)$ for *one* j Calculating $\theta_{f,s}(\boldsymbol{\sigma}^z)$ for *one* (f,s)	$\propto N$		
Calculating $\theta_j(\boldsymbol{\sigma}^z)$ for *all* j Calculating $\theta_{f,s}(\boldsymbol{\sigma}^z)$ for *all* (f,s)	$\propto \alpha N^2$		$\propto 8\alpha N^2$
Calculating cosh products (with θs pre-computed)	$\propto \alpha N$		$\propto 8\alpha N$
Dominant complexity to compute $\Psi(\boldsymbol{\sigma}^z)$	$O(\alpha N^2)$		

Table A.1: Computational cost and complexity of evaluating $\Psi(\boldsymbol{\sigma}^z)$.

A.1 Complexity of VMC

One VMC optimization step can be split in two parts: (i) generating N_{samples} independent samples, (ii) updating the N_{params} variational parameters with or without stochastic reconfiguration (SR).

A.1.1 Sampling.

One Markov Chain Monte Carlo step involves selection of a new configuration and acceptance of it. Starting from a configuration $\boldsymbol{\sigma}^z$, the next one in the Markov chain is chosen such that $\langle \boldsymbol{\sigma}^z | H | \boldsymbol{\sigma'}^z \rangle \neq 0$, i.e. they are related by an Hamiltonian element. In the $K_1 - K_2$ model in chapter 3, $\boldsymbol{\sigma}^z$ is connected to $O(N)$ configurations on average.

Calculating the probability of accepting $\boldsymbol{\sigma'}^z$ involves computing the ratio $\Psi(\boldsymbol{\sigma'}^z)/\Psi(\boldsymbol{\sigma}^z)$. It is in fact possible to compute it efficiently by taking advantage of the fact that $\boldsymbol{\sigma}^z$ and $\boldsymbol{\sigma'}^z$ are related by a Hamiltonian move, which changes the value of 4 spins for all kinetic terms in the $K_1 - K_2$ model. For two configurations differing by only one spin flip at site k, i,e. ${\sigma'}_k^z = -\sigma_k^z$, we have the following relations ($\sigma_i^z \in \{-1, +1\}$) for a non-symmetric RBM:

$$\theta_j(\boldsymbol{\sigma'}^z) = \theta_j(\boldsymbol{\sigma}^z) - 2W_{kj}\sigma_k^z \tag{A.5}$$

and for a symmetric RBM:

$$\theta_{f,s}(\boldsymbol{\sigma'}^z) = \theta_{f,s}(\boldsymbol{\sigma}^z) - 2W_k^{(f)}\widetilde{\sigma_k^z}(s) \tag{A.6}$$

Therefore, we only need to compute once the value $\Psi(\boldsymbol{\sigma}^z)$ at the beginning of sampling, and by storing the values $\theta_j(\boldsymbol{\sigma}^z)$ or $\theta_{f,s}(\boldsymbol{\sigma}^z)$, we only need to apply the above $O(1)$ update to the $O(\alpha N)$ cosh terms to compute the $\Psi(x_i)$. This allows to drop a factor of N in the complexity.

We often consider $O(N)$ local moves as being the unit of Monte Carlo steps, i.e sweeps. One must also keep in mind that autocorrelation between samples can increase the number of sweeps needed to get independent samples by a possibly large prefactor. All in all, the complexity of generating N_{samples} independent samples is $O(\alpha N^2 \times N_{\text{samples}})$ in all cases.

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Selecting $\boldsymbol{\sigma'}^z$	$O(N)$		
Updating $\theta_j(\boldsymbol{\sigma}^z)$ for *all* j Updating $\theta_{f,s}(\boldsymbol{\sigma}^z)$ for *all* (f,s)	$\propto \alpha N$		$\propto 8\alpha N$
Calculating cosh products (with θs pre-computed)	$\propto \alpha N$		$\propto 8\alpha N$
Dominant cost of computing $\Psi(\boldsymbol{\sigma'}^z)/\Psi(\boldsymbol{\sigma}^z)$	$O(\alpha N)$		

Table A.2: Computational cost and complexity for sampling in VMC.

A.1.2 Updating variational parameters.

Updating the value of a variational parameter is done according to:

$$\theta_{k+1} = \theta_k - \lambda \sum_{k'} S^{-1}_{k,k'} \frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_{k'}} \tag{A.7}$$

This means one has to calculate (i) the energy derivative $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$ with respect to all variational parameters θ_k and (ii) the inverse of the S matrix when stochastic reconfiguration is used. These quantities can be estimated with the following equations:

$$\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k} = 2\Re\left[\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} e_L^*(x_i)\left(O_k(x_i) - \overline{O}_k\right)\right] \tag{A.8}$$

$$S_{k,k'} = \Re\left[\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} (O_k(x_i) - \overline{O}_k)(O_{k'}(x_i) - \overline{O}_{k'})\right] \tag{A.9}$$

We see that computing the above expressions amounts to calculating the wave function logarithmic derivatives $O_k(x_i)$ with respect to all the variational parameters and all the Monte Carlo samples x_i, their average over the samples $\overline{O_k}$ and the local energies $e_L(x_i)$ for all samples $\{x_i\}$.

Computing $O_k(\boldsymbol{\sigma}^z)$ and $\overline{O_k}$. Note that here k refers to the αN^2 variational parameters (W_{ij}) of the rRBM, and to αN variational parameters $(W_i^{(f)})$ of the $\mathcal{T}$-invariant rRBM and C_{4v}-invariant rRBM. The above-mentioned quantities are defined as follows for non-symmetric RBMs:

$$O_k(\boldsymbol{\sigma}^z) \equiv \frac{\partial \log \Psi}{\partial W_{ij}}(\boldsymbol{\sigma}^z) = \sigma_i^z \tanh\left[\theta_j(\boldsymbol{\sigma}^z)\right] \tag{A.10}$$

and for symmetric RBMs:

$$O_k(\boldsymbol{\sigma}^z) \equiv \frac{\partial \log \Psi}{\partial W_i^{(f)}}(\boldsymbol{\sigma}^z) = \sum_{s=1}^{S} \widetilde{\sigma_i^z}(s) \tanh\left[\theta_{f,s}(\boldsymbol{\sigma}^z)\right] \tag{A.11}$$

$$\overline{O_k} \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} O_k(x_i) \tag{A.12}$$

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Calculating $\widetilde{\sigma_i^z}(s)$ for all i and s	-	$\propto N^2$	$\propto 8N^2$
Calculating $\theta_j(\boldsymbol{\sigma}^z)$ for *all* j Calculating $\theta_{f,s}(\boldsymbol{\sigma}^z)$ for *all* (f,s)	$\propto \alpha N^2$		$\propto 8\alpha N^2$
Calculate $O_k(\boldsymbol{\sigma}^z)$ for *one* k (with θs pre-computed)	$O(1)$	$\propto N$	$\propto 8N$
Calculate $O_k(\boldsymbol{\sigma}^z)$ for *all* k (with θs pre-computed)	$\propto \alpha N^2$		$\propto 8\alpha N^2$
Dominant cost to compute $O_k(\boldsymbol{\sigma}^z)$ for all samples	$O(\alpha N^2 \times N_{\text{samples}})$		
Dominant cost to compute $\overline{O_k}$ for all k (with all $\overline{O_k}(x_i)$ pre-computed)	$O(\alpha N^2 \times N_{\text{samples}})$	$O(\alpha N \times N_{\text{samples}})$	

Table A.3: Computational cost and complexity for the calculations of the wave function derivatives in VMC.

Local energies $e_L(x)$. The local energies are calculated as follows:

$$e_L(\boldsymbol{\sigma}^z) = \sum_{\boldsymbol{\sigma'}^z} \langle \boldsymbol{\sigma}^z | H | \boldsymbol{\sigma'}^z \rangle \frac{\Psi(\boldsymbol{\sigma'}^z)}{\Psi(\boldsymbol{\sigma}^z)} \tag{A.13}$$

It is possible to use the same trick as in Sec. A.1.1 since $\boldsymbol{\sigma}^z$ and $\boldsymbol{\sigma'}^z$ are related by an Hamiltonian move. We compute once $\Psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^z)$ (which amounts to $O(\alpha N^2)$ as shown in Tab. A.1) and computing the value $\Psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma'}^z)$ only require $O(\alpha N)$ operations (see Tab. A.2). Since there are $O(N)$ non-zero elements in the column of the $K_1 - K_2$ Hamiltonian, calculating all $\langle \boldsymbol{\sigma}^z | H | \boldsymbol{\sigma'}^z \rangle \Psi(\boldsymbol{\sigma'}^z)$ terms scales as $O(\alpha N^2)$. The dominant cost of computing $e_L(\boldsymbol{\sigma}^z)$ on all samples is then $O(\alpha N^2 \times N_{\text{samples}})$.

Stochastic reconfiguration. Let us estimate the cost of inverting the S matrix, given that the values of $O_k(x_i)$ and $\overline{O_k}$ have already been computed. As noted in [Carleo & Troyer 2017], due to the symmetric nature of S and its product structure, it is not necessary to explicitly evaluate each of its elements and then compute its inverse. Iterative solvers can reduce the cost greatly as showed below:

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Calculating *one* element $S_{k,k'}$ (with $O_k(x_i)$ and $\overline{O_k}$ precomputed)	$O(N_{\text{samples}})$		
Calculating the full S matrix	$O(\alpha^2 N^4 \times N_{\text{samples}})$	$O(\alpha^2 N^2 \times N_{\text{samples}})$	
Inverting S	$O(\alpha^3 N^6)$	$O(\alpha^3 N^3)$	
Iterative solving of S^{-1}	$O(\alpha N^2 \times N_{\text{samples}})$	$O(\alpha N \times N_{\text{samples}})$	

Table A.4: Computational complexity of the stochastic reconfiguration algorithm.

Dominant cost for parameter update. Once the quantities $O_k(x_i)$, $\overline{O_k}$, $e_L(x_i)$, $S^{-1}_{k,k'}$ are computed, the energy derivative in Eq. A.8 can be evaluated in $O(N_{\text{samples}})$ and the RBM parameters updated.

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Calculating $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$ for all k (with $O_k(x_i)$, $\overline{O_k}$, $e_L(x_i)$ precomputed)	$O(\alpha N^2 \times N_{\text{samples}})$	$O(\alpha N \times N_{\text{samples}})$	
Updating θ_k RBM parameters for all k without SR (with $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$ precomputed)	$O(\alpha N^2)$	$O(\alpha N)$	
Updating θ_k with SR (with $\frac{\partial E_{\boldsymbol{\theta}}}{\partial \theta_k}$, $S^{-1}_{k,k'}$ precomputed)	$O(\alpha^2 N^4)$	$O(\alpha^2 N^2)$	

Table A.5: Computational cost and complexity for

In summary, the dominant cost $\boxed{O(\alpha N^2 \times N_{\text{samples}})}$ is equally distributed over the different algorithm sections. Though we expect slower execution for non-symmetric RBMs when using stochastic reconfiguration. We note a computational cost greater by a factor of 8 for the C_{4v}-invariant ansatz compared to the $\mathcal{T}$-invariant ansatz, that shows up whenever the wave function is evaluated.

A.1.3 Computing energy and observables.

The local energy is already computed during the optimization procedure, so the variational energy can be estimated with a $O(N_{\text{samples}})$ cost. The computation of $S(\pi, \pi)$, $P(0, \pi)$ or σ scale as $O(N \times N_{\text{samples}})$.

A.2 RQMC with guiding RBMs

Let us recall that selection probability t and acceptance probability a are governed by the following equations:

$$t^{(+1)}(\mathcal{R}'|\mathcal{R}) = p_{x_T,x_0} \quad , \quad a^{(+1)}(\mathcal{R}'|\mathcal{R}) = \min\left[1, \frac{b_{x_0}}{b_{x_{p-1}}}\right] \tag{A.14}$$

$$t^{(-1)}(\mathcal{R}'|\mathcal{R}) = p_{x_T,x_p} \quad , \quad a^{(-1)}(\mathcal{R}'|\mathcal{R}) = \min\left[1, \frac{b_{x_p}}{b_{x_1}}\right] \tag{A.15}$$

Let us remind that $b_x = \sum_{x'} \mathcal{G}_{x,x'}$ and $p_{x',x} = \mathcal{G}_{x,x'}/b_x$ and

$$\mathcal{G}_{x,x'} = \langle x|(-H)|x'\rangle \frac{\Psi(x')}{\Psi(x)} \tag{A.16}$$

A.2.1 Sampling.

The reptile is initialized with p configurations all connected between each other by a Hamiltonian element. We first start by evaluating $p_{x_T,x}$, b_x, for all configuration x of the reptile and all x_T configurations connected to x. The selection and acceptance of a new configuration x_{new} can be computed from quantities p_{x_T,x_0}, b_{x_0}, etc.. that we know the value. Once a new configuration x_{new} has been selected and accepted, we need to compute the associated weights $b_{x_{\text{new}}} = \sum_{x'} \mathcal{G}_{x_{\text{new}},x'}$ and probabilities $p_{x',x_{\text{new}}} = \mathcal{G}_{x_{\text{new}},x'}/b_{x_{\text{new}}}$. Since all pairs x_{new}, x' are always related with an Hamiltonian move, we can use the same trick as in Sec. A.1.1.

Operation	rRBM	$\mathcal{T}$-invariant rRBM	C_{4v}-invariant rRBM
Pick a configuration x_{new} at random from distributions p_{x_T,x_0} or p_{x_T,x_p}	$O(N)$		
Calculating $\mathcal{G}_{x_{\text{new}},x'}$ for one x'	$\propto \alpha N$		$\propto 8\alpha N$
Calculating $\mathcal{G}_{x_{\text{new}},x'}$ for all x'	$\propto \alpha N^2$		$\propto 8\alpha N^2$
Calculating $\mathcal{G}_{x_{\text{new}},x'}$ for all x' *without guide*	$O(N)$		
Calculating $b_{x_{\text{new}}}$ (with $\mathcal{G}_{x_{\text{new}},x'}$ precomputed)	$O(N)$		
Calculating all $p_{x',x_{\text{new}}}$ (with $\mathcal{G}_{x_{\text{new}},x'}$ precomputed)	$O(N)$		

Table A.6: Computational cost and complexity of sampling in RQMC.

The dominant cost of one reptile move is $O(N)$ without a guide and $O(\alpha N^2)$ for all guides considered (although in practice the prefactor may be small). Moreover it is common to scale a Monte Carlo sweep with p Monte Carlo updates (even though an equally natural choice could be to scale sampling with systems size N, but we did not do that). In summary, the dominant cost without a guide is $O(N \times p \times N_{\text{samples}})$ whereas it is $O(\alpha N^2 \times p \times N_{\text{samples}})$ with all guides considered (though we note a computational cost prefactor of 8 for C_{4v}-invariant rRBM). In addition to that, equilibration of the Markov chain sometimes requires to discard of the order of 10% of the N_{samples}.

A.2.2 Computing energy and observables.

The local energy is obtained for free since $e_L(x) = -b_x$ and the ground state energy can be estimated with a $O(N_{\text{samples}})$ cost. The computation cost of $S(\pi, \pi)$, $P(0, \pi)$ or σ scales as $O(N \times N_{\text{samples}})$.

Appendix B

Résumé en français

La physique de la matière condensée a pour objet l'étude de modèles mathématiques censés décrire des matériaux réels. Ces modèles "jouets" sont conçus pour être assez riches pour rendre compte des propriétés de systèmes physiques réels, tout en étant assez simples pour pouvoir être traité analytiquement ou numériquement. Les laboratoires sont le lieu de la découverte de nouveaux faits expérimentaux qui intriguent et peuvent résister longtemps aux efforts de compréhension des physiciens. Parmi les phénomènes bien connus de la matière condensée mais qui échappent encore aujourd'hui à une théorisation satisfaisante, on peut compter la supraconductivité à haute-température [Bednorz & Müller 1986], l'effet Hall quantique fractionnaire[Tsui *et al.* 1982, Laughlin 1983] et, apparue dernièrement, la supraconductivité qui est observée lorsque deux feuillets de graphène sont placés l'un sur l'autre avec un léger décalage angulaire [Cao *et al.* 2018].

Une branche relativement nouvelle de la physique concerne l'étude de modèles complexes en utilisant la puissance de calcul des ordinateurs. A mi-chemin entre les expériences en laboratoire et les calculs analytiques, la physique numérique concerne la simulation numérique de ces modèles de matériau. Une simplification utile pour l'approche numérique est de considérer des modèles sur réseau où les degrés de liberté sont disposés dans un espace discrétisé (ce qui se justifie en matière condensée étant donné que les atomes s'organisent de manière périodique dans l'espace pour former des cristaux). Le modèle de Hubbard [Hubbard & Flowers 1963] est central dans le domaine et décrit des électrons pouvant sauter d'un sommet à l'autre du réseau et interagissant via un terme de répulsion sur site qui modélise l'interaction de Coulomb sous-jacente:

$$\mathcal{H} = \underbrace{-t \sum_{\langle i,j\rangle,\sigma} c^{\dagger}_{i,\sigma} c_{j,\sigma} + \text{h.c.}}_{\text{electron hopping}} + \underbrace{U \sum_{i} n_{i,\uparrow} n_{i,\downarrow}}_{\text{Coulomb repulsion}} \tag{B.1}$$

où $\langle \cdots \rangle$ désigne les sites voisins d'un réseau de dimension d. L'opérateur fermionique $c^{\dagger}_{j,\sigma}$ ($c_{j,\sigma}$) crée (détruit) un électron avec un spin σ (*up* $\uparrow$ ou *down* $\downarrow$) sur une orbitale résidant sur le site j.

En général, les solutions exactes de modèle sur réseau sont rares au-delà de systèmes unidimensionnels, et habituellement les développements analytiques doivent en dernier recours s'appuyer sur des approximations ou des hypothèses qui ne peuvent pas être rigoureusement justifiées. Par conséquent, les études basées sur des simulations numériques permettent de contourner ces limitations. Nous présentons ci-dessous deux méthodes représentatives des techniques souvent utilisées pour étudier des systèmes quantiques fortement corrélés.

Diagonalisation exacte

La façon la plus directe d'étudier un modèle de spins quantiques est de construire la matrice du Hamiltonien et de la diagonaliser. Ceci donne accès en principe aux états propres exacts ainsi

qu'au spectre d'énergie pour tout système de taille finie. Cependant, à cause de la croissance exponentielle de la taille de l'espace de Hilbert avec la taille du système (la taille de l'espace de Hilbert de N spins-1/2 est 2^N), la matrice du Hamiltonien devient exponentiellement grande et en pratique la limite en temps ou en mémoire est rapidement atteinte pour une dizaine de spins. Néanmoins, cette méthode étant exacte par nature, elle est d'une grande utilité pour tester de nouvelles méthodes en utilisant les résultats exacts obtenus par diagonalisation sur des petites systèmes comme base de comparaison. Dans certains cas, une étude à taille finie accompagné d'une théorie d'échelle appropriée et d'une analyse des symétries est suffisante pour extrapoler des observables dans la limite thermodynamique et obtenir un diagramme de phase [Wietek *et al.* 2017].

Méthodes variationnelles : *compresser*

Une stratégie pour contourner la complexité exponentielle d'une fonction d'onde quantique est de la paramétrer par un faible nombre de paramètres. Plus précisément, pour un état quantique $|\Psi\rangle = \sum_i c_i |i\rangle$, le nombre de coefficients c_i croît exponentiellement avec la taille du système. Le but des approches variationnelles est d'encoder ces amplitudes de manière succincte avec un nombre de paramètres variationnels qui ne croît qu'en loi polynomiale de la taille du système. Par construction, cette approche est biaisée par la paramétrisation choisie, néanmoins des approximations pertinentes d'états fortement corrélés peuvent être trouvées.

Étant donnés un Hamiltonien H et son état fondamental $|E_0\rangle$, comment pouvons-nous approximer $|E_0\rangle$ avec une certaine classe de fonctions d'onde ? Une manière de procéder est d'utiliser le principe variationnel qui donne une relation entre l'énergie de tout état variationnel $|\Psi_{\text{var}}\rangle$ et l'énergie de l'état fondamental E_0 :

$$E_0 \leq E_{\text{var}} = \frac{\langle \Psi_{\text{var}} | \mathcal{H} | \Psi_{\text{var}} \rangle}{\langle \Psi_{\text{var}} | \Psi_{\text{var}} \rangle} \tag{B.2}$$

En conséquence, l'énergie de tout état variationnel $|\Psi_{\text{var}}\rangle$ fournit une borne supérieure à la valeur exacte de l'énergie du fondamental. Sans connaître E_0, plus l'énergie E_{var} est basse, plus on s'attend à ce que l'état $|\Psi_{\text{var}}\rangle$ soit proche du véritable état fondamental. En pratique, on cherchera donc à optimiser les paramètres variationnels pour minimiser E_{var}, ceci peut être effectué avec des techniques d'échantillonnage Monte Carlo.

Apprentissage automatique et matière condensée

Le domaine de la physique n'échappe pas à la tendance générale actuelle qui a vu la quantité de données exploser. Comme la capacité de calcul et de mémoire augmentent et sont plus accessibles, les ensembles de données deviennent plus grands et les échantillons de meilleure qualité, comme on a pu le voir en physique des hautes énergies et la quantité croissante de données collectées par le LHC ou les grands télescopes. Par ailleurs, il existe des similarités entre la physique et le domaine de l'apprentissage automatique. On peut noter par exemple le parallèle entre la complexité exponentielle d'un système quantique en interaction avec la *malédiction de la dimension* qui affecte l'apprentissage automatique et stipule que le nombre de données d'entraînement doit croître exponentiellement vite avec la dimension des échantillons comme montré sur la figure B.1.

Figure B.1: Quand la dimension des données d'entrée augmente (de gauche à droite), le volume de l'espace des données augmente exponentiellement. Cela signifie que la taille des données d'entraînement doit croître exponentiellement avec la dimension en entrée pour qu'une machine puisse être entraînée avec un échantillon représentatif de données. Figure issue de [Goodfellow *et al.* 2015].

Les systèmes étudiés en matière condensée et les données en apprentissage automatique sont souvent caractérisés par leurs corrélations et leur symétries. Par exemple, il a été montré que les corrélations entre pixels [Ruderman 1994] ou entre mots [Ebeling & Pöschel 1994] décroissent en loi de puissance ce qui rappelle les corrélations dans les systèmes physiques proche d'une transition de phase. Dans le même esprit, les symétries aident souvent à la compréhension des systèmes quantiques comme elles peuvent réduire l'effort numérique de certaines tâches d'apprentissage.

Bien connu dans la théorie statistique de l'apprentissage et utilisé très tôt sur des grands ensembles de données issues d'expériences de physique des particules, l'apprentissage automatique n'est devenu un nouveau phénomène en physique de la matière condensée que très récemment. Il a d'abord été appliqué à la détermination des énergies d'atomisation de molécules [Rupp *et al.* 2012, Pilania *et al.* 2013] ou à l'accélération de simulations Monte Carlo [Huang & Wang 2017, Liu *et al.* 2017a, Xu *et al.* 2017]. Les échanges entre les deux domaines sont réciproques quand par exemple des réseaux de tenseurs sont utilisés pour classifier des images [Stoudenmire & Schwab 2016] ou encore quand la compréhension du fonctionnement des réseaux neurones se base sur des concepts empruntés au groupe de renormalisation [Mehta & Schwab 2014]. Sur le plan expérimental, des progrès notables ont été observés pour la tomographie d'état quantique [Torlai *et al.* 2018], la préparation d'états [Bukov *et al.* 2018] ou l'estimation de paramètres [Greplova *et al.* 2017] en utilisant des techniques d'apprentissage automatique.

Dans les sections suivantes, après avoir introduit la structure d'un réseau de neurones, nous résumons les trois chapitres de ma thèse qui sont centrés sur trois applications différentes de l'apprentissage automatique à la physique de la matière condensée.

B.1 Réseaux de neurones

Le premier exemple de réseau de neurones a été introduit dans les années 40 sous la forme de perceptrons [Rosenblatt 1957] et conçu à l'origine comme une possible modélisation du cerveau. De nos jours, la première motivation s'est effacée au profit d'une utilisation centrée sur des tâches de classification ou de génération de données comme des images, des sons ou des vidéos.

Les réseaux de neurones sont composés de neurones qui ont des propriétés qui ressemblent

(de loin) à des neurones réels. Pour résumer, les neurones artificiels sont des fonctions simples qui prennent plusieurs valeurs en entrée et renvoie une sortie après des opérations non-linéaires. Un neurone est défini de la manière suivante :

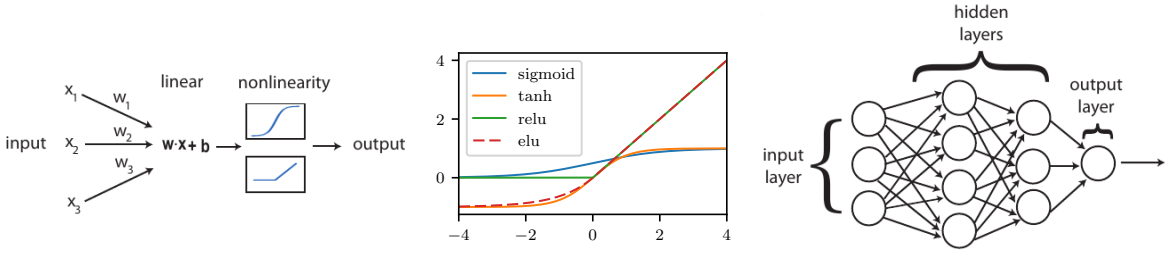
$$\text{Neuron}(\mathbf{x}) = f(\mathbf{W}^T\mathbf{x} + b) \tag{B.3}$$

où f est une fonction d'activation non-linéaire comme visible sur la figure B.2(b), $\mathbf{W}$ et b sont respectivement des vecteurs de poids et un biais scalaire, et $\mathbf{x}$ est un vecteur réel de données d'entrée. Par conséquent, la sortie d'un neurone est le résultat d'une transformation affine sur le vecteur d'entrée suivie d'une opération non-linéaire f.

Les réseaux de neurones, comme leur nom l'indique, sont construits en connectant des neurones ensemble. Les neurones sont habituellement organisés en couches successives (voir figure B.2(c)). Une couche est un ensemble de neurones qui va agir simultanément sur les valeurs d'entrée. L'équation ci-dessus peut être modifiée pour prendre la forme :

$$\text{Layer}(\mathbf{x}) = f(\mathbf{W}^T\mathbf{x} + \mathbf{b}) \tag{B.4}$$

où dans ce cas la fonction f est appliquée à chacun des éléments du vecteur d'entrée.



(a) (b) (c)

Figure B.2: (a) Décomposition d'un neurone, les parties linéaire and non-linéaires sont représentées, extrait de [Mehta *et al.* 2019]. (b) Fonctions d'activations les plus utilisées dans les applications d'apprentissage profond. (c) Architecture standard d'un réseau de neurone *feed-forward*, extrait de [Mehta *et al.* 2019].

Apprentissage profond. Un réseau de neurone qui est obtenu en empilant plusieurs couches de neurones est dit *profond*. Le vif intérêt suscité par les réseaux de neurones profonds vient du fait qu'ils sont particulièrement adaptés pour représenter des concepts qui s'organisent en une hiérarchie de concepts de haut-niveau obtenus en composant des concepts de bas-niveau. Ceci explique le succès de ces modèles appliqués à du texte, du son ou des images. Par exemple, détecter un chat dans une image signifie qu'un algorithme doit apprendre le concept du chat, qui est lui-même obtenu comme la composition d'autres concepts comme le concept de queue ou d'oreilles, qui peuvent à nouveau s'exprimer comme des compositions de concepts primaires comme des formes ou des textures. La figure B.3 montre comment le concept d'une personne peut être décrit par des éléments de bas-niveau, qui sont ensuite combinés entre eux dans les couches supérieures permettant à la fin de détecter des concepts de plus en plus abstraits ou de grande échelle et assigner un label à une image.

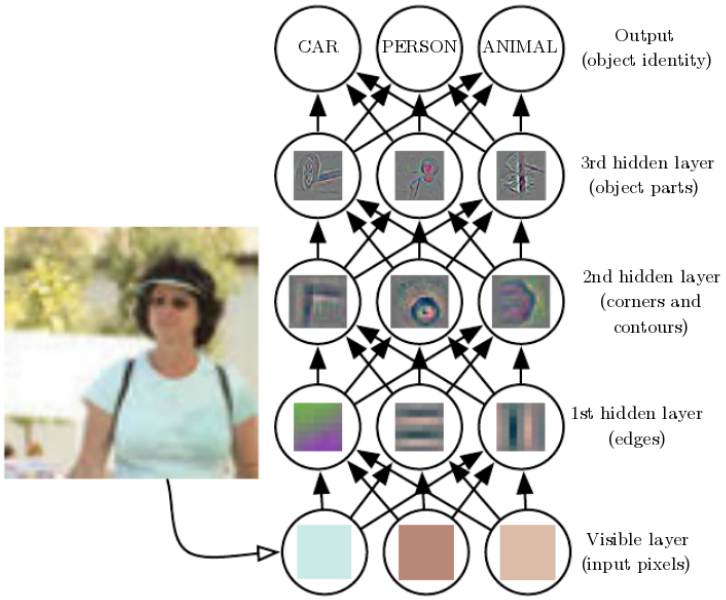


Figure B.3: Illustration schématique d'un réseau de neurones profond dont la sortie est obtenue en combinant des concepts de complexité croissante. Les premières couches sont capables de détecter des propriétés comme des gradients de couleur ou des petits motifs, ensuite les couches supérieures combinent ces détails pour permettre la détection de concepts de plus grande échelle comme des formes ou des objets entiers jusqu'à renvoyer la description la plus abstraite de l'image, c'est-à-dire le label. Image issue de [Goodfellow *et al.* 2016].

B.2 Détection automatique de phases de la matière

Une des premières applications de l'apprentissage automatique à la matière condensée a été de traiter le problème de classification de phases de la matière. Traditionnellement, la compréhension des phases et l'obtention de diagrammes de phases est réalisé après avoir identifié un paramètre d'ordre, qui peut être une observable physique prenant différentes valeurs dans chaque phase. Bien que ces diagnostics soient connus dans des cas simples comme le modèle d'Ising, la grande majorité du temps il est difficile de connaître le paramètre d'ordre, son existence n'étant d'ailleurs pas tout le temps garantie. De nouvelles perspectives sur ce sujet ont émergé à la suite d'une série de travaux [Wang 2016, Carrasquilla & Melko 2017, Wetzel 2017, van Nieuwenburg *et al.* 2017, Broecker *et al.* 2017a] dans lesquels un algorithme apprend à classifier des phases de la matière de manière autonome. Cette approche s'applique aussi bien à des données expérimentales qu'issues de simulations numériques et ouvre la voie, par exemple, à la découverte de nouveaux paramètres d'ordre.

B.2.1 Méthode supervisée pour détecter une transition

Carrasquilla et Melko ont proposé une méthode basée sur de l'apprentissage supervisé [Carrasquilla & Melko 2017] qui fonctionne comme suit : (i) la compréhension physique du modèle est utilisée pour labelliser des échantillons d'entraînement dans des limites bien comprises du diagramme de phase (voir la zone rouge dans la figure B.4), (ii) ensuite un réseau

de neurones est entraîné pour bien classifier les échantillons d'entraînement selon leur phase, (iii) enfin le réseau entraîné est utilisé pour prédire la phase correspondant à des échantillons provenant de tout l'espace des phases, ces prédictions sont enfin utilisées pour tracer le diagramme de phase.

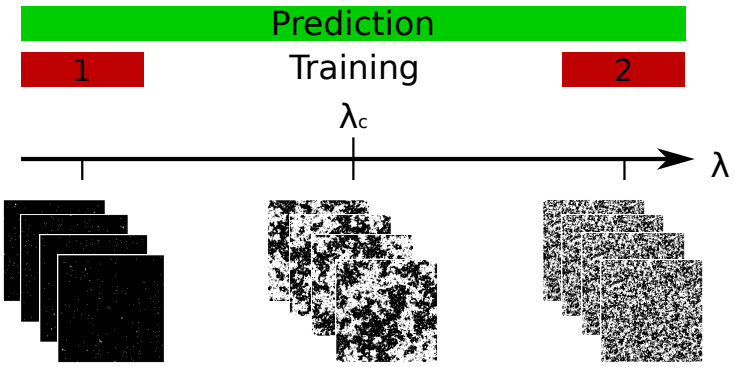


Figure B.4: L'approche supervisée fonctionne en entraînant un réseau de neurones à distinguer des échantillons des phases 1 et 2, la compréhension physique du système est utilisée pour labeller des données issues de limites bien comprises du diagramme de phase. La prédiction des phases se fait sur d'autres échantillons obtenus dans tout l'espace des phases.

Carrasquilla et Melko ont montré que, entraînés sur des configurations de spins classiques avec un label ferromagnétique (basse température) et paramagnétique (haute température) générées par des simulations Monte Carlo du modèle d'Ising sur réseau carré, un réseau de neurones est capable de classifier correctement les données d'entraînement mais également est capable de prédire de façon précise la température critique ainsi que l'exposant critique ν lié à la longueur de corrélation connus exactement. De manière assez remarquable, ils montrent aussi que ce réseau de neurones entraîné sur le réseau carré peut aussi prédire précisément le comportement critique du même modèle sur réseau triangulaire.

B.2.2 Application à la transition ETH-MBL en 1d

Les limitations actuelles des méthodes numériques pour aborder le problème de la localisation à N corps (MBL pour *many-body localization*) et les applications prometteuses de l'apprentissage automatique en physique ont été à l'origine de travaux utilisant ces techniques dans des systèmes présentant des phases MBL. En particulier, les débats touchant à l'existence d'un paramètre d'ordre dans les transitions ETH-MBL (ETH pour *eigenstate thermalization hypothesis*) offre une occasion intéressante de tester la capacité des techniques d'apprentissage automatique à traiter une transition de phase difficile.

En conséquence, un nombre important de travaux ont porté sur des transitions ETH-MBL en une dimension dans des systèmes quantiques désordonnés et ont pu (i) montrer des résultats qualitativement en adéquation avec des approches plus conventionnelles et (ii) parfois détecter de nouvelles phases dans des modèles connus [Venderley *et al.* 2018, Hsu *et al.* 2018]. Bien que satisfaisants, nous trouvons que ces travaux ont les limitations suivantes :

- la détection d'une transition de phase est parfois affirmée en considérant des petits systèmes sans qu'une analyse en taille finie soit tentée,

- l'entraînement est réalisé sur des données très transformées comme le spectre d'intrication, ce qui pourrait biaiser l'apprentissage,
- l'influence des hyperparamètres n'est pas systématiquement étudiée,
- l'interprétation des réseaux de neurones est parfois absente,
- l'existence d'une première estimation du point de transition ($h_c \approx 3.7$ dans le modèle étudié) peut biaiser certains travaux et les pousser à juger positivement leurs résultats dès que cette valeur est retrouvée.

Notre travail a consisté à combler ces lacunes et à fournir une étude de la transition ETH-MBL assistée par des techniques d'apprentissage automatique en suivant les préceptes suivants :

- *Minimiser l'intervention humaine* pour réduire le biais qui pourrait venir de la façon dont les données sont prétraitées, du choix de l'architecture du réseau de neurones, de l'interprétation *ad hoc* de la sortie du réseau de neurones, etc.. Nous voulons rester aussi agnostique que possible en ce qui concerne la physique en jeu, ce qui permettrait de révéler de nouveaux phénomènes dans cette transition. En corollaire, plus les données d'entrée sont génériques, plus notre protocole pourra être appliqué à d'autres transitions de phase.
- *Accessibilité des grandes tailles*, la dimension des échantillons d'entrée doit être réduite pour que l'entraînement des réseaux de neurones reste possible pour les plus grandes tailles de système ($L = 24$) d'autant plus que nous voulons faire une analyse à taille finie.
- *Minimiser l'influence des paramètres non-physiques*, la variabilité des résultats en fonction des paramètres de la méthode (formatage des données, hyperparamètres) doit être étudiée et éliminée autant que possible.
- *Interpretabilité* : les données et l'architecture du réseau de neurones doivent rendre possible une interprétation de la fonction de décision apprise.

Ces directives peuvent parfois être antagonistes, les sections qui suivent présentent le compromis auquel nous avons abouti pour satisfaire au mieux ces contraintes.

B.2.3 Données

De nombreux types de données ont été considérés dans les précédents travaux : le spectre d'intrication d'états propres [van Nieuwenburg *et al.* 2017, Hsu *et al.* 2018, Schindler *et al.* 2017, Venderley *et al.* 2018, Durr & Chakravarty 2019], des observables dynamiques [Doggen *et al.* 2018, van Nieuwenburg *et al.* 2018], le spectre d'énergie [Rao 2018, Kausar *et al.* 2020] ou les états propres entiers [Zhang *et al.* 2019a, Huembeli *et al.* 2019]. C'est cette dernière approche que nous suivons puisqu'elle a l'avantage de n'opérer aucune compression de l'information, toutes les observables d'un système pouvant être calculées à partir d'un état propre. Faire ce choix implique cependant de traiter le problème de la dimensionnalité des échantillons. En effet, un état propre $|\Psi\rangle$ peut s'écrire dans la base des S^z noté $|i\rangle$ comme $|\Psi\rangle = \sum_i c_i|i\rangle$ et le nombre de coefficients $p_i \equiv |c_i|^2$ croît exponentiellement avec la taille du système. Notre solution a été de garder uniquement les N_c plus grandes probabilités p_i comme montré sur la figure B.5.

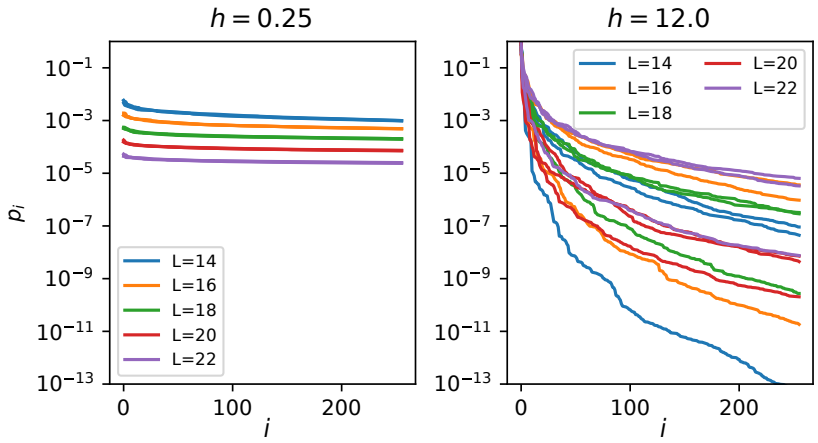


Figure B.5: Exemples des $N_c = 256$ plus grandes probabilités p_i pour des états propres dans le milieu du spectre pour différentes réalisations du désordre et tailles de système pour deux valeurs de désordre correspondant aux phases ETH (gauche) et MBL (droite).

Données. Nous obtenons des états propres exacts dans le milieu du spectre (température infinie) à l'aide de la méthode *shift-invert* [Pietracaprina *et al.* 2018b]. Pour les données d'entraînement, nous avons utilisé 1000 réalisations de désordre par force de désordre et 250 pour la prédiction pour des tailles allant de $L = 14$ à $L = 24$.

Réseau de neurones. Une architecture de réseau de neurones de faible profondeur (voir figure B.6) permet de satisfaire les contraintes liées à l'interprétabilité des prédictions, puisque le nombre de paramètres du réseau de neurones est raisonnablement faible.

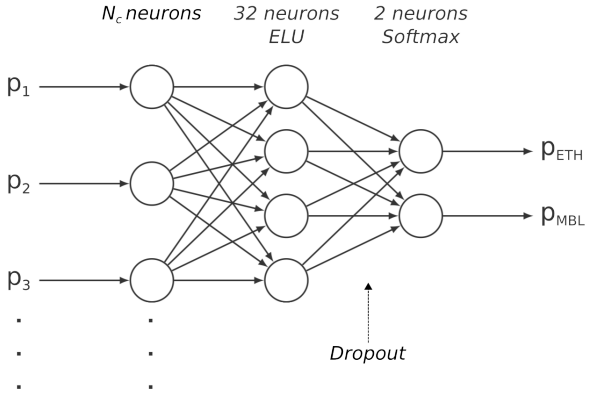


Figure B.6: Architecture du réseau de neurones que nous avons utilisée. Chaque état propre est donnée au réseau comme un vecteur de taille N_c et le réseau de neurones renvoie un nombre p_{MBL} dans $[0, 1]$ (avec $p_{\mathrm{MBL}} + p_{\mathrm{ETH}} = 1$) qui peut être interprété comme la probabilité de classer l'échantillon comme MBL.

B.2.4 Procédure uni-taille

Dans cette procédure, un ensemble de réseaux de neurones est entraîné séparément sur différentes tailles de système. Un désordre critique à taille finie $h_c(L)$ peut être déduit des prédictions du réseau de neurone, enfin nous obtenons le désordre critique dans la limite thermodynamique $h_c(\infty)$ et ν avec l'ansatz montré sur la figure ci-dessous.

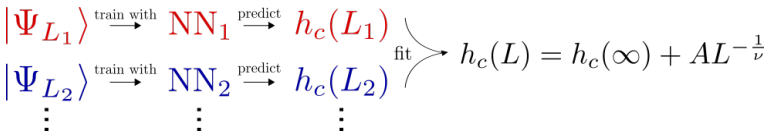


Figure B.7: Procédure uni-taille.

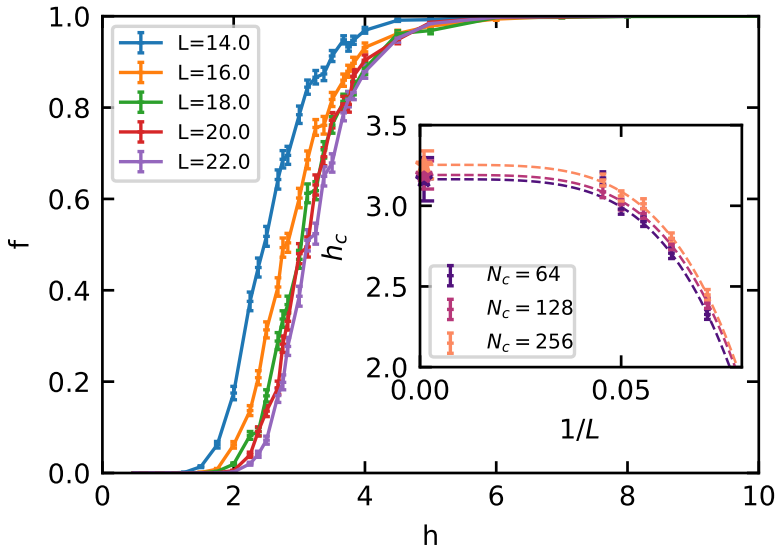


Figure B.8: Fraction des états propres classifiés comme MBL en fonction de la force de désordre pour la procédure uni-taille. Les prédictions sont moyennées sur 250 réalisations de désordre pour chaque force de désordre (avec 100 états propres par réalisation) et 50 entraînements indépendants. La troncature vaut $N_c = 256$. Les barres d'erreur indiquent l'erreur statistique due à l'échantillonnage du désordre. Encart : étude à taille à finie de $h_c(L)$ défini comme $f(h_c(L)) = 0.5$ pour différentes troncatures N_c.

Nous pouvons faire plusieurs remarques sur la figure B.8 : d'une part l'existence d'une région entièrement ETH (où tous les échantillons sont classifiés comme ETH) qui s'étend de $h = 0$ à $h = 2$ et une région entièrement MBL qui commence à partir de $h = 6$ pour toutes les tailles de système considérées. Un autre point important est l'organisation respective des courbes en fonction de la taille du système, la transition entre ETH et MBL a lieu à des désordres plus élevés à mesure que la taille augmente. Ce comportement est en accord avec d'autres observables (comme les statistiques spectrales ou la variance de l'intrication) utilisées dans l'analyse standard du système [Luitz *et al.* 2015].

L'analyse en taille finie donne un désordre critique $h_c \approx 3.2$ qui est plus faible que l'estimation habituelle plutôt autour de $h_c \approx 3.7$ [Luitz *et al.* 2015] et des valeurs de $\nu \approx 0.22$ très petites, qui semblent non-physiques.

Même si nos résultats montrent des comportements cohérents quand on varie N_c ou L, la procédure uni-taille autorise a priori la possibilité que des réseaux de neurones entraînés à des tailles différentes puissent apprendre différentes observables physiques. En effet, l'entraînement dépend par exemple de la capacité du réseau de neurones (nombre de couches, de neurones cachés) par rapport à la complexité des données d'entraînement (qui varie d'une taille à l'autre).

B.2.5 Procédure multi-taille

Le formatage des états propres tronqués à N_c coefficients permet d'utiliser un unique réseau de neurones pour traiter sur le même plan des données venant de différentes tailles de système (voir figure B.9). Cela pourrait aider le réseau de neurones à apprendre des caractéristiques des états propres qui ne sont pas spécifiques d'une taille donnée. Dans la suite, nous étudions les prédictions d'un réseau de neurones entraîné avec des données venant de tailles $L = 16, 18, 20, 22$.

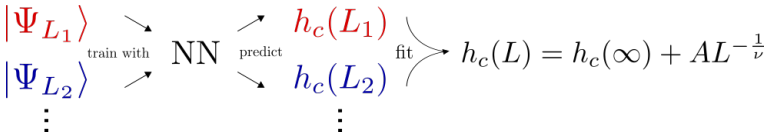


Figure B.9: Procédure multi-taille

La figure B.10 montre des similarités avec la figure B.7 mais dans ce cas les courbes correspondantes à différentes tailles de système sont plus espacées dans la région de transition. Notons que nous avons effectué une prédiction non-triviale en utilisant le réseau de neurones entraîné sur les tailles $L = 16, 18, 20, 22$ sur les tailles $L = 14$ et $L = 24$ qu'il n'a pas vues durant l'entraînement.

L'étude en taille finie aboutit à un résultat problématique car les prédictions dépendent fortement de la zone de l'espace des phases utilisée pour l'entraînement alors même que pour $h > 8$ on peut considérer le système comme ayant des propriétés homogènes et très MBL. En conséquence, les estimations de h_c varient entre $h_c \approx 4$ et $h_c \approx 6$ ce qui dépasse la valeur de référence et ν varie entre 0.6 et 1.4.

La procédure multi-taille était a priori plus à même de produire des résultats plus fiables, c'est-à-dire qui seraient moins sensibles aux effets de taille finie. Cependant, nous avons trouvé que le point de transition prédit dépend fortement de la région de l'espace des phases utilisée pour l'entraînement. Ceci est clairement une limitation de notre procédure puisque nous aimerions obtenir des valeurs critiques qui ne dépendent pas des données utilisées en entrée.

L'analyse des poids du réseau de neurones a aussi révélé que l'entraînement aboutit à une règle de classification qui est une moyenne des règles de classification trouvées en section précédente. Plus précisément, c'est la même fonction de décision qui aurait été obtenue en entraînant le réseau avec des données provenant de la taille $L = \frac{16+18+20+22}{4}$.

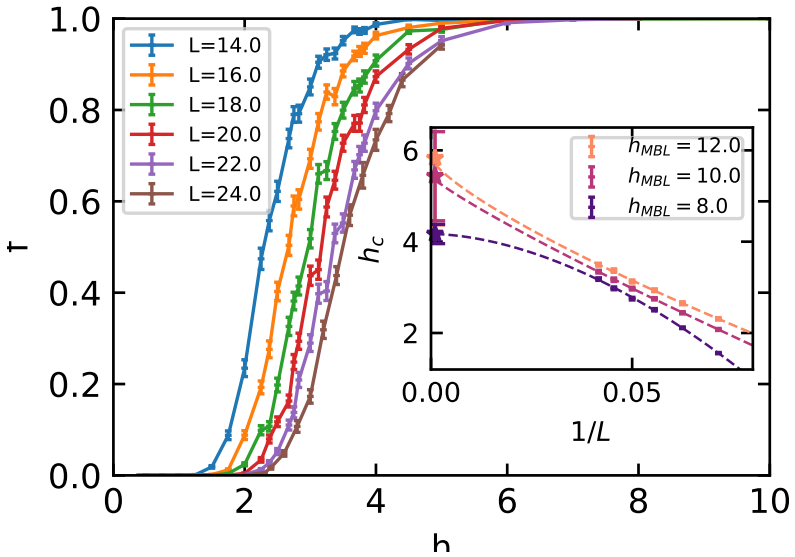


Figure B.10: Fraction des états propres classifiés comme MBL en fonction de la force de désordre pour la procédure multi-taille. Les prédictions sont moyennées sur 250 réalisations de désordre pour chaque force de désordre (avec 100 états propres par réalisation) et 50 entraînements indépendants. La troncature vaut $N_c = 256$. Les barres d'erreur indiquent l'erreur statistique due à l'échantillonnage du désordre. Encart : étude à taille à finie en utilisant des données labellées ETH à $h_{\mathrm{ETH}} = 0.25$ et labellées MBL à $h_{\mathrm{MBL}} = 8.0, 10.0, 12.0$.

B.2.6 Conclusion

Nos résultats montrent des tendances cohérentes lorsque l'on change la dimension des données d'entrée (la classification est stable avec la troncature N_c) et la taille physique du système considéré (le critère de classification s'adapte à la décroissance des premiers coefficients p_i) pour la procédure uni-taille.

L'objectif initial de notre travail était de faire une étude à taille finie d'un modèle présentant des phases MBL et ETH en utilisant des réseaux de neurones. Notre analyse révèle de nombreuses difficultés: l'analyse en taille finie est très sensible aux hyperparamètres du réseau de neurones (le choix des fonctions d'activation, l'ajout ou non de *dropout*). En plus de cela, il n'existe pas de critère inhérent à la méthode permettant de discriminer ces différents choix et d'une certaine manière, on peut voir notre analyse comme une sorte d'exploration de modèles (différentes machines de performance équivalente ont différentes manières de résoudre la même tâche) plutôt qu'une sélection de modèles (sélectionner la machine qui est la plus performante).

Les limitations viennent également de la dépendance en les données d'entraînement, nous avons montré que les prédictions du réseau de neurones et l'analyse à taille finie changent selon la zone de l'espace des phases choisie pour l'entraînement. De plus, quand les données d'entraînement contiennent des fonctions d'onde provenant de différentes tailles de système, le réseau de neurones a tendance à extraire des caractéristiques moyennées. En incluant une contrainte pour limiter cet effet (sous la forme d'un composant adversaire), la situation s'améliore au prix d'un plus grand nombre d'hyperparamètres à ajuster et d'un risque accru de biaiser un

peu plus les estimations finales.

Ces limitations apparaissent alors même que nous avons donné les meilleures données d'entrée (i) donnant directement les fonctions d'onde compressées de façon contrôlée et (ii) aussi en termes de tailles de système (jusqu'à $L = 24$ ce qui est l'état de l'art actuel dans le contexte de la MBL). Néanmoins nous trouvons que les différents cas capturent des tendances à taille finie cohérentes avec un changement brusque autour de la région critique, cela en utilisant un nombre limité de réalisations de désordre. Cette dernière remarque souligne un des avantages des réseaux de neurones, ils permettent de diminuer le coût de calcul par rapport aux méthodes conventionnelles. Un autre point intéressant (évoqué dans l'appendice B de [Théveniaut & Alet 2019]), que nous avons découvert en inspectant les différentes contributions à la variance des prédictions est que la classification d'une fonction d'onde par un réseau de neurones est bien corrélée avec son entropie d'intrication.

L'analyse à taille finie aboutit à des valeurs de h_c et ν assez différentes des estimations habituelles: $h_c \simeq 3.2$ et $\nu \simeq 0.22$ pour le cas uni-taille, et $h_c \simeq 5-6$ et $\nu \simeq 1.2-1.5$ dans les cas multi-tailles. L'analyse à taille finie de la transition MBL est aussi connue pour être particulièrement difficile, les tailles accessibles par diagonalisation exacte sont considérées trop petites pour sonder le bon comportement proche de la transition [Khemani *et al.* 2017, Panda *et al.* 2020]. Nous ne trouvons pas que les techniques d'apprentissage automatique permettent d'améliorer la situation, en tout cas avec les données et les approches que nous avons choisies. En particulier, il n'y a aucune raison d'accorder plus de crédit aux prédictions des réseaux de neurones qu'aux résultats obtenus avec les approches usuelles. La tendance globale qui semble se dessiner pointe vers une phase ETH plus large que détectée auparavant, même si nous soulignons qu'aucune valeur de champ $h_c(L)$ (obtenue individuellement pour chaque taille de système L) n'excède la valeur de référence $h_c = 3.7$ dans les barres d'erreur.

Notre étude des effets de taille finie de cette transition de phase aboutit à la conclusion qu'il faut toujours être conscient des multiples biais qui proviennent de l'utilisation de réseaux de neurones, et que l'intérêt de la méthode pourrait être restreint à des prédictions qualitatives plutôt qu'à des estimations précises. Cette dernière remarque s'applique d'autant plus pour des transitions de phase dont la nature ou la classe d'universalité est encore inconnue ou débattue et/ou pour lesquelles les données d'entrée sont limitées (en termes de taille accessibles par exemple).

B.2.7 MBL en deux dimensions

Dans [Théveniaut *et al.* 2020], nous apportons de nombreuses preuves numériques en faveur de l'existence d'une phase localisée dans des systèmes à deux dimensions, qui est une question du domaine très débattue.

Nous avons trouvé qu'un réseau de neurones entraîné avec le spectre d'intrication d'états propres exacts est capable d'apprendre à distinguer les régimes ETH et MBL pour un modèle quantique de dimères. Les résultats sont en adéquation avec le comportement d'autres observables.

B.3 Des états quantiques paramétrés par des réseaux de neurones

Un champ entier d'études s'est ouvert à la suite de l'article fondateur de Carleo et Troyer [Carleo & Troyer 2017] dans lequel ils proposent de paramétrer les amplitudes d'une fonction d'onde quantique par un réseau de neurones. Pour un système de N particules avec des nombres quantiques $s_1, \ldots, s_N$, ces états dénommés NQS (pour *neural-network quantum states*) peuvent s'écrire

$$|\Psi_{\text{NQS}}\rangle = \sum_{s_1,\cdots,s_N} f(s_1, \ldots, s_N)|s_1, \ldots, s_N\rangle \tag{B.5}$$

où f est un réseau de neurones, c'est-à-dire une fonction prenant un vecteur de grande dimension en entrée et renvoyant un nombre complexe. Eq. (B.5) définit un ensemble très large d'états quantiques selon le type de réseau de neurones (à propagation avant, convolutifs, récurrents,..), leur architecture (nombre de couches, de neurones,..) et les valeurs de leurs paramètres internes.

Les auteurs de [Carleo & Troyer 2017] ont montré que les NQS peuvent être utilisés comme approximations variationnelles d'états quantiques fondamentaux et ont montré d'excellentes performances pour des modèles de spins en dimension 1 et 2. La complexité de cette représentation découle directement de l'architecture du réseau de neurones f sous-jacent, ce qui procure flexibilité et expressivité à cet ansatz car augmenter la largeur ou la profondeur de f améliore en général la qualité de l'approximation variationnelle.

B.3.1 Machines de Boltzmann restreinte

Le premier ansatz considéré par [Carleo & Troyer 2017] était basé sur une machine de Boltzmann restreinte (RBM). Les RBMs contiennent des neurones visibles et cachés couplés les uns aux autres comme le montre la figure B.11.

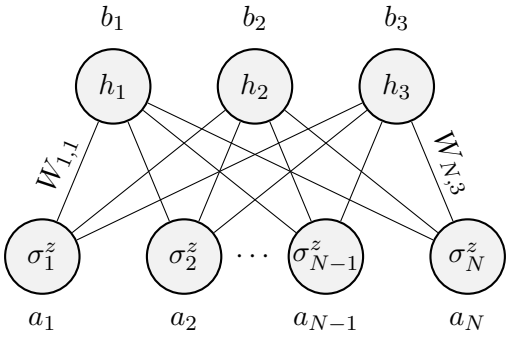


Figure B.11: Une RBM contenant N neurones visibles prenant en entrée les valeurs des magnétisations locales $\sigma_i^z = \pm 1$ d'un système de spins-1/2, et 3 neurones cachés $h_j = \pm 1$. Quelques poids sont montrés: les biais visibles a_i, les biais cachés b_j et la matrice de poids $W_{i,j}$.

Comme les neurones prennent des valeurs binaires en entrée, il est particulièrement naturel d'utiliser cette paramétrisation pour des systèmes ayant des degrés de liberté binaires comme un spin-1/2, pour lequel la magnétisation locale selon z $\sigma_i^z = \pm 1$ peut être choisie comme nombre quantique. Plus précisément, pour un système de N spins-1/2, tout état quantique $|\Psi\rangle$ peut

s'écrire dans la base des états $|\sigma_1^z\rangle \otimes \cdots \otimes |\sigma_N^z\rangle$ (avec $\hat{\sigma}_i^z|\sigma_i^z\rangle = \sigma_i^z|\sigma_i^z\rangle$). L'ansatz RBM consiste à paramétrer l'amplitude complexe de $|\Psi\rangle$ dans cette base, c'est-à-dire $\langle\sigma_1^z \cdots \sigma_N^z|\Psi\rangle$, par la fonction de partition d'une RBM comme suit:

$$\Psi_{\mathrm{RBM}}(\sigma_1^z, \ldots, \sigma_N^z) = \sum_{(h_1,\ldots,h_M)\in\{-1,1\}^M} \exp\left(\sum_{i=1}^{N} a_i\sigma_i^z + \sum_{j=1}^{M} b_j h_j + \sum_{i=1}^{N}\sum_{j=1}^{M} W_{ij}\sigma_i^z h_j\right) \tag{B.6}$$

où $\mathbf{a} = (a_i)_{i=1\ldots N} \in \mathbb{C}^N$, $\mathbf{b} = (b_j)_{j=1\ldots M} \in \mathbb{C}^M$ and $\mathbf{W} = (W_{ij})_{i=1\ldots N, j=1\ldots M} \in \mathbb{C}^{N\times M}$ sont des paramètres variationnels à valeur complexe. A noter que si ces paramètres étaient réels, la fonction d'onde $|\Psi\rangle$ serait réelle et positive et ne pourrait donc pas décrire un état fermionique possédant une structure de signe.

B.3.2 Le modèle $K_1 - K_2$

Dans [Paramekanti *et al.* 2002], Paramekanti, Balents et Fisher proposent l'existence d'un nouvel état de la matière bosonique dont les propriétés sont proches de celles de fermions. Les arguments théoriques mis en avant montrent la stabilité de cette phase si les termes cinétiques de type échange cyclique (voir figure B.12) sont suffisamment forts comparés aux termes habituels de saut entre proche voisins. Cette découverte est particulièrement intéressante pour l'étude du matériau solide 3-He [Roger 1983] et est possiblement reliée aux phénomènes de supraconductivité à haute température. De façon alternative, on peut voir la cinétique en échange cyclique comme un terme de saut d'une paire de particules – un boson et un trou formant un exciton – au travers du réseau, ce qui donne le nom de liquide de Bose excitonique (EBL). La phase EBL présente les propriétés thermodynamiques suivantes : une compressibilité non nulle, une chaleur spécifique en $\sim T\log(T)$ et l'existence de deux lignes d'excitations sans trou spectral dans l'espace de Brillouin, ce qui rappelle une phase liquide [Paramekanti *et al.* 2002].

S'inscrivant dans une série de tentatives infructueuses [Sandvik *et al.* 2002, Melko *et al.* 2004, Rousseau *et al.* 2004, Rousseau *et al.* 2005] de réaliser une phase EBL dans des modèles sur réseau, Tay et Motrunich [Tay & Motrunich 2010, Tay & Motrunich 2011] ont étudié un modèle de bosons de coeur dur avec deux types d'échanges cycliques, ils montrent des preuves numériques supportant l'existence de cette phase. Les opérateurs d'échange cycliques sur un réseau carré sont définis sur une plaquette $\mathbf{r}$ comme suit :

$$P_{\mathbf{r}}^{mn} = b_{\mathbf{r}}^\dagger b_{\mathbf{r}+m\hat{\mathbf{x}}} b_{\mathbf{r}+m\hat{\mathbf{x}}+n\hat{\mathbf{y}}}^\dagger b_{\mathbf{r}+n\hat{\mathbf{y}}} \tag{B.7}$$

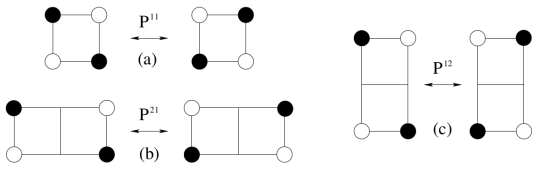


Figure B.12: Les opérateurs *ring-exchange* pour des plaquettes (a) 1×1, (b) 2×1 and (c) 1×2. Figure extraite de [Tay & Motrunich 2010].

Fig. B.12 montre trois types d'opérateurs plaquette qui apparaissent dans l'Hamiltonien considéré dans [Tay & Motrunich 2010, Tay & Motrunich 2011] et qui s'écrit:

$$H = -K_1 \sum_{\mathbf{r}} P_{\mathbf{r}}^{1\times 1} - K_2 \sum_{\mathbf{r}} \left(P_{\mathbf{r}}^{1\times 2} + P_{\mathbf{r}}^{2\times 1} \right) \tag{B.8}$$

En utilisant des méthodes variationnelles Monte Carlo (VMC) et Green function Monte Carlo (GFMC), Tay et Motrunich obtiennent le diagramme de phase montré figure B.13.

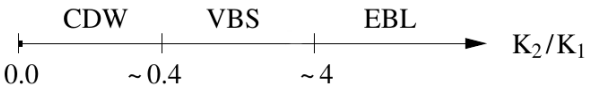


Figure B.13: Diagramme de phase du modèle B.8 obtenu par [Tay & Motrunich 2011]. Une onde de densité de charge (CDW) se développe à petit K_2, puis un cristal de liens de valence (VBS) est observé à des valeurs intermédiaires jusqu'à laisser place à une phase liquide de Bose excitonique qui s'étend au delà de $K_2/K_1 \approx 4$.

Les paramètres d'ordre de chaque phase sont les suivants:

1. **Facteur de structure** Une observable importante pour détecter une phase ordonnée est le facteur de structure densité, qui est défini comme la transformée de Fourier des corrélations densité-densité:

$$S(q_x, q_y) = \frac{1}{L^2} \sum_{\mathbf{r},\mathbf{r}'} e^{i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')} \langle n_{\mathbf{r}} n_{\mathbf{r}'} - \bar{n} \rangle \tag{B.9}$$

   Pour détecter une onde de densité de charge (CDW pour *charge density wave* en anglais) à (π, π), la valeur du facteur de structure au moment $\mathbf{q} = (\pi, \pi)$ est un bon indicateur puisqu'il atteint un maximum pour une configuration de bosons de type Néel.

2. **Facteur de structure plaquette** De la même manière, le paramètre d'ordre correspondant au cristal de liens de valence à $(0, \pi)$ s'obtient à partir du facteur de structure plaquette défini comme suit:

$$P(q_x, q_y) = \frac{1}{L^2} \sum_{\mathbf{r},\mathbf{r}'} e^{i\mathbf{q}\cdot(\mathbf{r}-\mathbf{r}')} \left\langle \left(P_{\mathbf{r}}^{11}\right)^2 \left(P_{\mathbf{r}'}^{11}\right)^2 \right\rangle \tag{B.10}$$

   La valeur à $\mathbf{q} = (0, \pi)$ sert comme paramètre d'ordre de cette phase.

3. **L'observable "croix"** Les conditions de l'existence d'une phase EBL ont été obtenues [Paramekanti *et al.* 2002]: (i) il ne doit pas y avoir d'ordre de charge, ce qui peut se voir par l'absence de pic de Bragg dans la zone de Brillouin et (ii) le facteur de structure en densité doit présenter des singularités le long des lignes $(0, q_y)$ et $(q_x, 0)$, qui ont été identifiées théoriquement comme des marqueurs de l'existence d'une surface de Bose. La condition (ii) peut être étudiée via l'opérateur "croix" défini dans [Tay & Motrunich 2011]:

$$\sigma(q_x, q_y) = \frac{S(q_x, q_y)}{4|\sin(q_x/2)\sin(q_y/2)|} \tag{B.11}$$

où $S_{\mathrm{EBL}}(q_x, q_y) \equiv 4|\sin(q_x/2)\sin(q_y/2)|$ est la prédiction de la théorie EBL [Paramekanti *et al.* 2002]. Comme mentionné dans [Tay & Motrunich 2011], la phase EBL est signalée si la valeur de σ proche des singularités reste finie dans la limite thermodynamique puisque cela signifie que $S(q_x, q_y) \underset{q_x \to 0}{\propto} S_{\mathrm{EBL}}(q_x, q_y)$ (ou de façon équivalente quand $q_y \to 0$). De plus, [Tay & Motrunich 2011] argumente basé sur une théorie des champs effective que la phase EBL n'est stable que dans le cas où σ reste au-dessus d'une valeur seuil $\sigma \geq \sigma_c = 3/16$. En pratique, nous évaluerons $\sigma(q_{\min} = 2\pi/L, q_y)$ ou de manière équivalente $\sigma(q_x, q_{\min} = 2\pi/L)$.

Motivés par l'explosion des travaux autour des NQS, notre travail tente d'étendre la validité des résultats de [Tay & Motrunich 2010], c'est-à-dire de confirmer ou infirmer leurs conclusions pour des tailles de système plus grands (jusqu'à $L \leq 16$ comparé à $L \leq 12$ dans [Tay & Motrunich 2010]) avec une étude variationnelle utilisant des NQS et des méthodes de projection guidée. Cela constitue un test pour l'efficacité des NQS dans ce diagramme des phases contenant plusieurs phases.

B.3.3 Résultats variationnels

Bien que les fonctions d'onde RBMs ne puissent pas décrire efficacement certains états de la matière [Gao & Duan 2017], leur performance demeure excellente dans une grande majorité des cas étudiés et fait partie des NQS les plus simples. Nous avons testé la capacité des RBMs à décrire les états fondamentaux du modèle (B.8) en comparant l'énergie et d'autres observables (comme les paramètres d'ordre présentés plus haut) de fonctions d'onde RBMs optimisées aux valeurs exactes obtenues par diagonalisation pour $L = 6$ (le maximum faisable) et aux résultats non biaisés donnés par les méthodes projectives (voir section suivante) pour $L = 12$. De plus, nous évaluons différents ansatz RBMs implémentant plus ou moins de symétries du modèle.

Nos résultats montrent que la précision des états variationnels RBMs est très bonne dans les trois phases pour $L = 6$ et $L = 12$, atteignant des erreurs relatives sur l'énergie de l'ordre de 10^{-3} respectivement 10^{-2} dès $\alpha = 4$. Le diagramme de phase montré en figure B.14 est établi en faisant une étude à taille finie des paramètres d'ordre des phases CDW et VBS et en analysant l'opérateur "croix" à grand K_2.

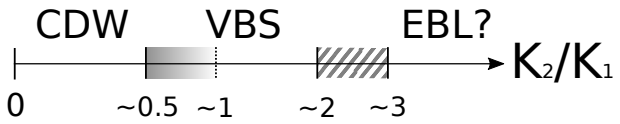


Figure B.14: Diagramme de phase VMC obtenu avec une fonction d'onde RBM invariante par translation avec $\alpha = 10$ et une étude en taille finie pour des systèmes de taille atteignant $L = 16$. Le premier gradient de couleur indique la coexistence des phases CDW et VBS. La zone hachurée indique une région où ni les corrélations plaquette ne sont détectées ni la stabilité de la phase EBL est établie par notre critère.

B.3.4 Diagramme de phase exact

Dans la section précédente, même si notre approche purement variationnelle avec un ansatz RBM est plutôt précise pour $L = 6$ ou $L = 12$, il est parfois difficile d'établir avec certitude l'extension

des différentes phases. Cela vient du fait que les tendances à taille finie des paramètres d'ordre sont difficiles à lire, de plus il est important de rappeler que les méthodes variationnelles sont biaisées par nature. Cette section est donc dévolue à nos résultats obtenus via des méthodes projectives non-biaisées. Ces méthodes peuvent être accélérées en utilisant des fonctions d'onde guide, ici des RBM précédemment optimisées. Les résultats sont présentés en figure B.15 et le diagramme de phase exact sur la figure B.16.

B.3.5 Conclusion

Nous avons établi que les fonctions d'onde RBM sont capables de bien approximer les phases solides et liquides du modèle $K_1 - K_2$. En particulier, augmenter le nombre de neurones cachés (α) permet d'améliorer la précision de l'ansatz de manière systématique. En comparaison de l'ansatz variationnel de [Tay & Motrunich 2010] basé sur une théorie d'onde de spins, les RBMs sont non seulement capables d'être plus précis sur l'énergie et les observables pertinentes même pour les plus petits α considérés et sans symétrie implémentée, mais semblent aussi capables de capturer des corrélations de type plaquette dans la limite thermodynamique, permettant de détecter la phase VBS. Nous avons aussi noté des améliorations en implémentant les symétries du modèle directement dans la RBM. Cependant, les effets de taille finie des paramètres d'ordre calculés en VMC restent difficiles à lire dans certaines parties de l'espace des phases, ce qui empêche de déterminer avec précision les limites des phases.

Une étude exacte du modèle $K_1 - K_2$ est possible à l'aide de Monte Carlo quantique "reptile" [Baroni & Moroni 1999]. Nous avons montré que la présence d'une fonction d'onde guide RBM était bénéfique sur bien des aspects (thermalisation plus courte, convergence plus rapide, barres d'erreur plus petites). Notre analyse de taille finie avec les méthodes de projection guidée par la RBM $\alpha = 10$ invariante par translation a confirmé le diagramme de phase trouvé dans [Tay & Motrunich 2010], et étend leurs résultats à $L \leq 16$. En guise d'avertissement, nous voulons souligner que nous ne pouvons pas exclure la possibilité que pour de meilleurs guides ou des tailles plus grandes, le critère de stabilité sur l'opérateur "croix" pourrait ne pas être satisfait. Néanmoins, comme évoqué dans [Tay & Motrunich 2010], ce critère pourrait ne pas être l'argument final pour conclure sur la stabilité de la phase EBL.

En ce qui concerne les futurs développements, nous travaillons actuellement à trouver des indices de l'existence de la phase EBL en étudiant l'intrication. En effet, comme montré dans [Lai *et al.* 2013], l'entropie d'intrication se comporte en $\sim L \log(L)$ pour des systèmes de bosons avec des surfaces de Bose, ce qui fournit un possible indicateur pour la phase EBL. Alors que l'entropie d'intrication est très difficile à obtenir en Monte Carlo quantique, il est plus facile de la calculer efficacement en VMC [Hastings *et al.* 2010] en particulier pour des fonctions d'onde RBM.

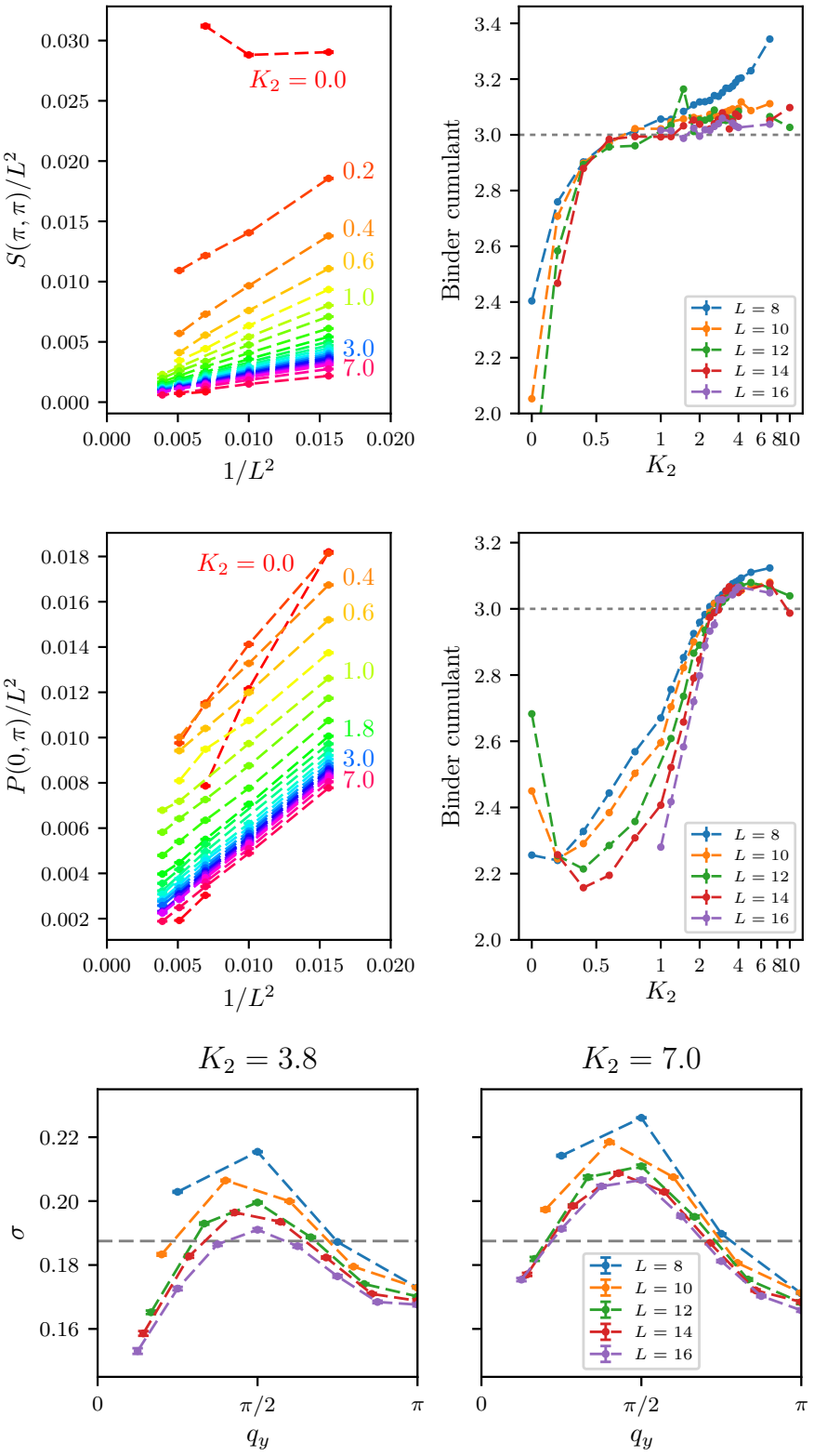


Figure B.15: Résultats RQMC. (Haut gauche) $S(\pi,\pi)/L^2$ en fonction $1/L^2$ pour différentes valeurs de K_2. (Haut droit) Cumulant de Binder de $S(\pi,\pi)$ pour des systèmes allant de $L = 8$ à $L = 16$. L'échelle est linéaire pour $K_2 \in [0,1]$ et logarithmique pour $K_2 \in [1,10]$. (Milieu gauche) $P(0,\pi)/L^2$ en fonction de $1/L^2$ pour différentes valeurs de K_2. (Milieu droit) Cumulant de Binder de $P(0,\pi)$ pour des systèmes allant de $L = 8$ à $L = 16$. (Bas) Opérateur "croix" à grand K_2 et différentes tailles de système. Les tirets indiquent le seuil limite de stabilité de la phase EBL $\sigma_c = 3/16$.

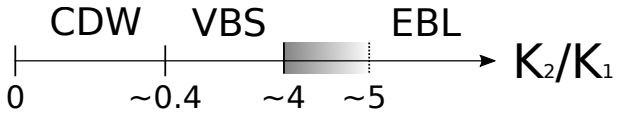


Figure B.16: Diagramme de phase RQMC obtenu avec une fonction d'onde guide RBM invariante par translation et $\alpha = 10$, l'étude à taille finie est effectuée sur des tailles allant jusqu'à $L = 16$. La zone grisée correspond à une région dans laquelle la phase VBS n'est plus présente et où la stabilité de la phase EBL n'est pas visible dans la limite des tailles étudiées.

B.4 Correction d'erreur dans les codes quantiques à l'aide de techniques d'apprentissage par renforcement

Dans le contexte du développement des ordinateurs quantiques, la réalisation de qubits robustes passe inévitablement par la mise en place de stratégies de correction d'erreurs permettant de protéger l'information stockée dans le qubit. Les codes quantiques topologiques sont récemment apparus comme des candidats prometteurs pour l'implémentation de qubits robustes [Kitaev 2003]. En encodant un qubit logique dans les propriétés topologiques d'un système plus grand (qui peut être composé de plusieurs spins physiques), l'effet de la décohérence (changement d'état d'un spin individuel par exemple) peut être corrigé activement.

B.4.1 Le code torique

Le code torique [Kitaev 2003] est un modèle de spins-1/2 vivant sur les arêtes d'un réseau carré avec des conditions périodiques au bord. L'Hamiltonien s'écrit :

$$H = -\sum_v A_v - \sum_p B_p \tag{B.12}$$

qui inclut des stabilisateurs "sommet" $A_v = \Pi_{i\in v}\sigma_i^x$ où v est un sommet du réseau (le produit est réalisé sur les spins qui sont les arêtes les plus proches en rouge sur la figure B.17) et des stabilisateurs "plaquette" (le produit est réalisé sur toutes les arêtes de la plaquette en vert sur la figure B.17). Les opérateurs A_v et B_v ont pour valeurs propres ± 1, commutent mutuellement et avec le Hamiltonien. L'état fondamental du code torique se situe dans le secteur où $A_v = +1$ et $B_v = +1$ pour tous les stabilisateurs "sommet" et "plaquette". Nous pouvons montrer que l'état fondamental est 4 fois dégénéré, ce qui permet de définir deux qubits logiques.

Les excitations de plus faible énergie qui projettent le système en dehors de l'état fondamental s'obtiennent en appliquant un opérateur de Pauli sur l'état fondamental $|\mathrm{GS}\rangle$. L'état excité $\sigma_i^x|\mathrm{GS}\rangle$ produit une erreur de type *bit-flip*, de manière similaire $\sigma_j^z|\mathrm{GS}\rangle$ produit une erreur de type *phase-flip*. On considère dans la suite un modèle de bruit usuel où les erreurs surviennent avec une probabilité p sur chaque qubit physique. Une erreur σ_j^z survenant sur l'arête j change la valeur des stabilisateurs plaquette qui partage cette arête et donne $B_v = -1$. On définit un syndrome comme étant la position des stabilisateurs prenant la valeur -1. Contrairement aux opérateurs de Pauli, les syndromes peuvent être mesurés sans altérer l'état du système (puisque B_v commute avec H), ce qui donne un accès indirect aux erreurs. La difficulté centrale de la correction d'erreur vient du fait que la relation entre syndromes et erreurs physiques n'est pas bijective, en d'autres termes plusieurs configurations d'erreurs peuvent donner le même syndrome. L'ambiguïté inhérente des syndromes ne permet pas d'éviter une erreur logique d'apparaître après correction.

B.4.2 Décoder avec des algorithmes évolutionnistes

L'algorithme NEAT (pour *NeuroEvolution Augmented Topologies*) a été introduit par Stanley et Miikulainen dans [Stanley & Miikkulainen 2002] et fait partie de la famille d'algorithmes évolutionnistes. Ces algorithmes fonctionnent en faisant évoluer une population d'individus selon des heuristiques inspirées par l'évolution biologique. A chaque génération, la performance de chaque individu est évaluée et la population de la génération suivante est obtenue par mutation,

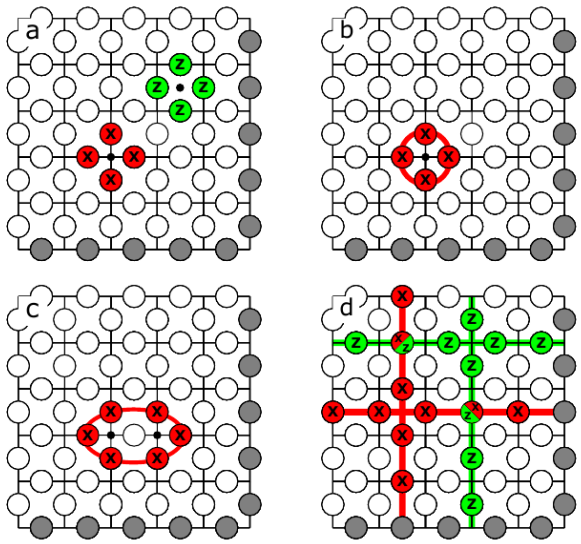


Figure B.17: (a) Action des opérateurs stabilisateurs A_v et B_p. (b) L'action d'un opérateur sommet peut être vu comme l'ajout d'une boucle triviale. (c) Une boucle plus grande de topologie triviale. (d) Les quatre boucles d'opérateurs σ^x et σ^z de topologie non-triviale qui permettent de définir les quatre états fondamentaux. La figure est issue de [Andreasson *et al.* 2019].

reproduction ou sélection des meilleurs individus. Les mutations consistent à changer de manière aléatoire certaines propriétés des individus. La sélection est basée sur la performance individuelle et seulement les individus les plus adaptés survivent. La reproduction permet de mélanger les gènes de deux individus pour créer une descendance qui héritent de leurs traits respectifs. Grâce à ces heuristiques, les individus s'adaptent de plus en plus à leur environnement à mesure que l'évolution perdure.

Dans l'algorithme NEAT, la population est composée de réseaux de neurones. La spécificité de cet algorithme est que les mutations n'altèrent pas uniquement les poids du réseau de neurones mais également son architecture. En effet, des neurones ou des connexions entre neurones peuvent être ajoutés ou supprimés.

Dans le contexte de la correction d'erreur dans les codes quantiques, un réseau de neurones prend en entrée la donnée des syndromes B_v, c'est-à-dire un vecteur binaire de taille L^2 et renvoie la position de l'opérateur σ^x à appliquer, c'est-à-dire un vecteur de taille $2L^2$ qui peut être réduit à 4 en utilisant l'invariance par translation du modèle. La performance de chaque réseau de neurones est évaluée en lui présentant une série de "puzzles" (configurations d'erreurs aléatoires) à corriger, son score correspond à la fraction de puzzles dont la correction n'a pas introduit d'erreur logique.

B.4.3 Résultats

La performance d'un algorithme de correction d'erreurs peut se mesurer par la proportion d'erreur logique en fonction du taux de *bit-flip*. Fig. B.18 montre que les résultats obtenus sont similaires à l'algorithme MWPM: les performances se détériorent à mesure que le taux de *bit-flip* augmente et un point de croisement des courbes est visible autour de $p_c \approx 0.08 - 0.09$

pour NEAT, ce qui est un peu moins bien que $p_c \approx 0.1$ pour MWPM, mais la fidélité critique est meilleure : ≈ 0.85 pour NEAT comparé à ≈ 0.75 pour MWPM. De plus, nous pouvons noter que la fidélité logique est plus élevée que MWPM pour $p \geq 0.1$. Malgré la simplicité de notre approche, nous sommes capables d'atteindre des niveaux de performance similaires aux approches antérieures basées sur des techniques d'apprentissage automatique [Torlai & Melko 2017, Andreasson *et al.* 2019] (sur le même code et le même type de bruit), surpassant MWPM dans les régimes les plus bruités.

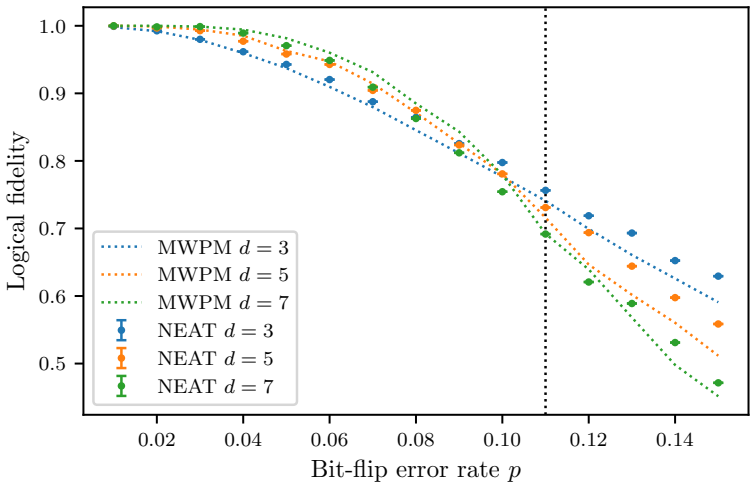


Figure B.18: Fidélité logique en fonction du taux d'erreur physique p pour différentes tailles de codes. La performance de MWPM est montrée par des lignes pointillées et les symboles montrent la performance du meilleur réseau de neurones trouvé par l'algorithme NEAT après quelques centaines de générations. La ligne verticale en pointillée montre la borne supérieure théorique du seuil critique d'erreur [Dennis *et al.* 2002]. L'évaluation est faite sur 5000 échantillons indépendants et aléatoires pour chaque taux d'erreur physique.

Ces résultats ont été obtenus avec un coût de calcul bien moindre comparés aux travaux précédents et des réseaux de neurones peu profonds. Les réseaux de neurones possèdent 1000 à 10000 fois moins de paramètres que les réseaux profonds utilisés en *Q-learning*. Ceci est une preuve forte que des réseaux de neurones profonds ne sont pas nécessaires pour accomplir ces tâches de correction d'erreur. Plusieurs conséquences découlent de cela: (i) des décodeurs d'exécution rapide seront probablement nécessaires dans des applications réelles, (ii) nous nous attendons à une meilleure mise à l'échelle des ressources de calcul pour les plus grandes tailles, enfin (iii) la faible complexité des réseaux de neurones permet a priori une compréhension plus aisée.

Un autre avantage de notre approche est la possibilité de décoder des grands codes avec des réseaux de neurones qui ont été au départ entraînés sur des petits codes. De cette manière, l'apprentissage des stratégies de décodage pour les grands codes peut être accéléré en initialisant la population de réseaux de neurones par le meilleur réseau de neurone trouvé pour la taille de code inférieur.

Bibliography

[Abadi *et al.* 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/. (Cited on page 29.)

[Abanin & Papić 2017] Dmitry A. Abanin and Zlatko Papić. Recent progress in many-body localization. *Annalen der Physik*, 529(7):1700169, 2017. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.201700169. (Cited on page 44.)

[Abanin *et al.* 2019a] D. A. Abanin, J. H. Bardarson, G. De Tomasi, S. Gopalakrishnan, V. Khemani, S. A. Parameswaran, F. Pollmann, A. C. Potter, M. Serbyn, and R. Vasseur. Distinguishing localization from chaos: challenges in finite-size systems, 2019. arXiv:1911.04501. (Cited on page 45.)

[Abanin *et al.* 2019b] Dmitry A. Abanin, Ehud Altman, Immanuel Bloch, and Maksym Serbyn. Colloquium: Many-body localization, thermalization, and entanglement. *Reviews of Modern Physics*, 91(2):021001, May 2019. URL: https://link.aps.org/doi/10.1103/RevModPhys.91.021001. (Cited on page 44.)

[Alet & Laflorencie 2018] Fabien Alet and Nicolas Laflorencie. Many-body localization: An introduction and selected topics. *Comptes Rendus Physique*, 19(6):498–525, Sep 2018. URL: http://www.sciencedirect.com/science/article/pii/S163107051830032X. (Cited on page 44.)

[Alet & Sørensen 2003] Fabien Alet and Erik S. Sørensen. Directed geometrical worm algorithm applied to the quantum rotor model. *Physical Review E*, 68(2):026702, August 2003. URL: https://link.aps.org/doi/10.1103/PhysRevE.68.026702. (Cited on page 6.)

[Alet *et al.* 2016] Fabien Alet, Kedar Damle, and Sumiran Pujari. Sign-Problem-Free Monte Carlo Simulation of Certain Frustrated Quantum Magnets. *Physical Review Letters*, 117(19):197203, November 2016. URL: https://link.aps.org/doi/10.1103/PhysRevLett.117.197203. (Cited on page 6.)

[Amari 1998] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. (Cited on pages 27 and 84.)

[Andreasson *et al.* 2019] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. Quantum error correction for the toric code using deep reinforcement learning.

Quantum, 3:183, Sep 2019. arXiv:1811.12338. (Cited on pages 109, 110, 112, 113, 117, 118, 119, 120, 127, 159 and 160.)

[Arute *et al.* 2019] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(77797779):505–510, Oct 2019. URL: https://www.nature.com/articles/s41586-019-1666-5. (Cited on pages 6 and 125.)

[Baireuther *et al.* 2018] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *Quantum*, 2:48, Jan 2018. arXiv:1705.07855. (Cited on page 112.)

[Baroni & Moroni 1999] Stefano Baroni and Saverio Moroni. Reptation quantum monte carlo: A method for unbiased ground-state averages and imaginary-time correlations. *Physical Review Letters*, 82(24):4745–4748, Jun 1999. URL: https://link.aps.org/doi/10.1103/PhysRevLett.82.4745. (Cited on pages 86, 87 and 155.)

[Basko *et al.* 2006] D. M. Basko, I. L. Aleiner, and B. L. Altshuler. Metal-insulator transition in a weakly interacting many-electron system with localized single-particle states. *Annals of Physics*, 321(5):1126–1205, May 2006. arXiv:0506617. (Cited on page 43.)

[Beach *et al.* 2018] Matthew J. S. Beach, Anna Golubeva, and Roger G. Melko. Machine learning vortices at the kosterlitz-thouless transition. *Phys. Rev. B*, 97:045207, Jan 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.97.045207. (Cited on pages 38, 42 and 64.)

[Becca & Sorella 2017] Federico Becca and Sandro Sorella. *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, 2017. URL: https://www.cambridge.org/core/books/quantum-monte-carlo-approaches-for-correlated-systems/. (Cited on pages 36, 81, 84, 88 and 93.)

[Bednorz & Müller 1986] J. G. Bednorz and K. A. Müller. Possible high T_c superconductivity in the Ba-La-Cu-O system. *Zeitschrift fur Physik B Condensed Matter*, 64:189–193, June 1986. doi:10.1007/BF01303701. (Cited on pages 3 and 139.)

[Bellemare *et al.* 2015] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 4148–4152. AAAI Press, 2015. (Cited on page 31.)

[Biamonte *et al.* 2017] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(76717671):195–202, Sep 2017. URL: https://www.nature.com/articles/nature23474. (Cited on page 127.)

[Bloch *et al.* 2012] I. Bloch, J. Dalibard, and S. Nascimbène. Quantum simulations with ultracold quantum gases. *Nature Physics*, 8:267–276, 2012. (Cited on page 6.)

[Bohrdt *et al.* 2019] Annabelle Bohrdt, Christie S. Chiu, Geoffrey Ji, Muqing Xu, Daniel Greif, Markus Greiner, Eugene Demler, Fabian Grusdt, and Michael Knap. Classifying snapshots of the doped hubbard model with machine learning. *Nature Physics*, 15(9):921–924, Sep 2019. arXiv:1811.12425. (Cited on page 42.)

[Broecker *et al.* 2017a] Peter Broecker, Fakher F. Assaad, and Simon Trebst. Quantum phase recognition via unsupervised machine learning, Jul 2017. arXiv:1707.00663. (Cited on pages 7, 41, 125 and 143.)

[Broecker *et al.* 2017b] Peter Broecker, Juan Carrasquilla, Roger G. Melko, and Simon Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Scientific Reports*, 7(1), Dec 2017. arXiv:1608.07848. (Cited on pages 38 and 43.)

[Bukov *et al.* 2018] Marin Bukov, Alexandre G. R. Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta. Reinforcement learning in different phases of quantum control. *Phys. Rev. X*, 8:031086, Sep 2018. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.031086. (Cited on pages 7, 127 and 141.)

[Calandra-Buonaura & Sorella 1998] Matteo Calandra-Buonaura and Sandro Sorella. Numerical study of the two-dimensional heisenberg model using a green function monte carlo technique with a fixed number of walkers. *Physical Review B*, 57(18):11446–11456, May 1998. URL: https://link.aps.org/doi/10.1103/PhysRevB.57.11446. (Cited on pages 86 and 90.)

[Canfield & McKay 2005] E. Rodney Canfield and Brendan D. McKay. Asymptotic enumeration of dense 0-1 matrices with equal row sums and equal column sums. *The Electronic Journal of Combinatorics*, page R29–R29, Jun 2005. URL: https://www.combinatorics.org/ojs/index.php/eljc/article/view/v12i1r29. (Cited on page 91.)

[Cao *et al.* 2018] Yuan Cao, Valla Fatemi, Shiang Fang, Kenji Watanabe, Takashi Taniguchi, Efthimios Kaxiras, and Pablo Jarillo-Herrero. Unconventional superconductivity in magic-angle graphene superlattices. *Nature*, 556(76997699):43–50, Apr 2018. URL: https://www.nature.com/articles/nature26160. (Cited on pages 3 and 139.)

[Carleo & Troyer 2017] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, Feb 2017. URL: http://science.sciencemag.org/content/355/6325/602. (Cited on pages 8, 77, 78, 80, 85, 126, 134 and 151.)

[Carleo *et al.* 2018] Giuseppe Carleo, Yusuke Nomura, and Masatoshi Imada. Constructing exact representations of quantum many-body systems with deep neural networks. *Nature Communications*, 9(11):1–11, Dec 2018. URL: https://www.nature.com/articles/s41467-018-07520-3. (Cited on page 80.)

[Carleo *et al.* 2019a] Giuseppe Carleo, Kenny Choo, Damian Hofmann, James E. T. Smith, Tom Westerhout, Fabien Alet, Emily J. Davis, Stavros Efthymiou, Ivan Glasser, Sheng-Hsuan Lin, Marta Mauri, Guglielmo Mazzola, Christian B. Mendl, Evert van Nieuwenburg, Ossian O'Reilly, Hugo Théveniaut, Giacomo Torlai, Filippo Vicentini, and Alexander

Wietek. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, page 100311, 2019. URL: http://www.sciencedirect.com/science/article/pii/S2352711019300974. (Cited on pages 93, 98 and 126.)

[Carleo *et al.* 2019b] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, Dec 2019. URL: https://link.aps.org/doi/10.1103/RevModPhys.91.045002. (Cited on page 3.)

[Carrasquilla & Melko 2017] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, Feb 2017. arXiv:1605.01735. (Cited on pages 7, 25, 39, 43, 52, 57, 125 and 143.)

[Carrasquilla *et al.* 2019] Juan Carrasquilla, Giacomo Torlai, Roger G. Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, Oct 2019. arXiv:1810.10584. (Cited on page 78.)

[Chamberland & Ronagh 2018] Christopher Chamberland and Pooya Ronagh. Deep neural decoders for near term fault-tolerant experiments. *Quantum Science and Technology*, 3(4):044002, Jul 2018. arXiv:1802.06441. (Cited on page 112.)

[Chandrasekharan & Li 2012] Shailesh Chandrasekharan and Anyi Li. Fermion bag solutions to some sign problems in four-fermion field theories. *Physical Review D*, 85(9):091502, May 2012. URL: https://link.aps.org/doi/10.1103/PhysRevD.85.091502. (Cited on page 6.)

[Chandrasekharan & Wiese 1999] Shailesh Chandrasekharan and Uwe-Jens Wiese. Meron-Cluster Solution of Fermion Sign Problems. *Physical Review Letters*, 83(16):3116–3119, October 1999. URL: https://link.aps.org/doi/10.1103/PhysRevLett.83.3116. (Cited on page 6.)

[Chen *et al.* 2018] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted boltzmann machines and tensor network states. *Physical Review B*, 97(8):085104, Feb 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.97.085104. (Cited on page 79.)

[Ch'ng *et al.* 2017] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami. Machine Learning Phases of Strongly Correlated Fermions. *Physical Review X*, 7(3):031038, August 2017. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.031038. (Cited on pages 42 and 67.)

[Chollet *et al.* 2015] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015. (Cited on page 29.)

[Choo *et al.* 2019] Kenny Choo, Titus Neupert, and Giuseppe Carleo. Study of the two-dimensional frustrated j1-j2 model with neural network quantum states. *Physical Review B*, 100(12):125124, Sep 2019. arXiv:1903.06713. (Cited on pages 25, 80 and 85.)

[Choromanska *et al.* 2015] Anna Choromanska, MIkael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *International Con-ference on Artificial Intelligence and Statistic*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 09–12 May 2015. PMLR. URL: http://proceedings.mlr.press/v38/choromanska15.html. (Cited on page 28.)

[Christie's 2018] Christie's. Is artificial intelligence set to become art's next medium?, 2018. URL: https://www.christies.com/features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx. (Cited on page 16.)

[Clark 2018] Stephen R. Clark. Unifying neural-network quantum states and correlator product states via tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 51(13):135301, Apr 2018. arXiv:1710.03545. (Cited on page 79.)

[Clevert *et al.* 2016] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), Feb 2016. arXiv:1511.07289. (Cited on page 53.)

[Cole *et al.* 2020] Alex Cole, Gregory J. Loges, and Gary Shiu. Quantitative and interpretable order parameters for phase transitions from persistent homology, 2020. arXiv:2009.14231. (Cited on page 126.)

[Cortes & Vapnik 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. URL: http://link.springer.com/10.1007/BF00994018. (Cited on page 22.)

[Cybenko 1989] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, page 12, 1989. URL: https://link.springer.com/article/10.1007/BF02551274. (Cited on page 23.)

[Dawid *et al.* 2020] Anna Dawid, Patrick Huembeli, Michał Tomza, Maciej Lewenstein, and Alexandre Dauphin. Phase detection with neural networks: Interpreting the black box, 2020. arXiv:2004.04711. (Cited on page 126.)

[De Roeck & Huveneers 2017] Wojciech De Roeck and Francois Huveneers. Stability and instability towards delocalization in many-body localization systems. *Phys. Rev. B*, 95:155129, Apr 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.155129. (Cited on page 71.)

[De Roeck & Imbrie 2017] Wojciech De Roeck and John Z. Imbrie. Many-body localization: stability and instability. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2108):20160422, 2017. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2016.0422. (Cited on page 71.)

[de Roeck *et al.* 2016] Wojciech de Roeck, Francois Huveneers, Markus Muller, and Mauro Schiulaz. Absence of many-body mobility edges. *Physical Review B*, 93(1):014203, Jan 2016. arXiv:1506.01505. (Cited on page 46.)

[De Tomasi *et al.* 2019] Giuseppe De Tomasi, Frank Pollmann, and Markus Heyl. Efficiently solving the dynamics of many-body localized systems at strong disorder. *Phys. Rev. B*, 99:241114, Jun 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.241114. (Cited on page 71.)

[Deng *et al.* 2017a] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Machine learning topological states. *Physical Review B*, 96(19):195145, Nov 2017. arXiv:1609.09060. (Cited on page 79.)

[Deng *et al.* 2017b] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Quantum entanglement in neural network states. *Physical Review X*, 7(2):021021, May 2017. arXiv:1701.04844. (Cited on pages 79 and 126.)

[Dennis *et al.* 2002] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43:4452–4505, Sep 2002. URL: http://adsabs.harvard.edu/abs/2002JMP....43.4452D. (Cited on pages 111, 120 and 160.)

[Doggen *et al.* 2018] Elmer V. H. Doggen, Frank Schindler, Konstantin S. Tikhonov, Alexander D. Mirlin, Titus Neupert, Dmitry G. Polyakov, and Igor V. Gornyi. Many-body localization and delocalization in large quantum chains. *Physical Review B*, 98(17):174202, Nov 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.98.174202. (Cited on pages 47 and 145.)

[Doggen *et al.* 2020] Elmer V. H. Doggen, Igor V. Gornyi, Alexander D. Mirlin, and Dmitry G. Polyakov. Slow many-body delocalization beyond one dimension, 2020. arXiv:2002.07635. (Cited on page 71.)

[Domingo Colomer *et al.* 2020] Laia Domingo Colomer, Michalis Skotiniotis, and Ramon Muñoz-Tapia. Reinforcement learning for optimal error correction of toric codes. *Physics Letters A*, 384(17):126353, Jun 2020. URL: http://dx.doi.org/10.1016/j.physleta.2020.126353. (Cited on pages 112, 113, 117, 119, 120 and 127.)

[Dong *et al.* 2014] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, page 184–199. Springer International Publishing, 2014. doi:10.1007/978-3-319-10593-2_13. (Cited on page 16.)

[Duchi *et al.* 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(null):2121–2159, July 2011. (Cited on pages 27 and 84.)

[Duclos-Cianci & Poulin 2010a] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Physical Review Letters*, 104(5):050504, Feb 2010. URL: https://link.aps.org/doi/10.1103/PhysRevLett.104.050504. (Cited on pages 110 and 127.)

[Duclos-Cianci & Poulin 2010b] Guillaume Duclos-Cianci and David Poulin. A renormalization group decoding algorithm for topological quantum codes, 2010. arXiv:1006.1362. (Cited on page 112.)

[Dumitrescu *et al.* 2019] Philipp T. Dumitrescu, Anna Goremykina, Siddharth A. Parameswaran, Maksym Serbyn, and Romain Vasseur. Kosterlitz-thouless scaling at many-body localization phase transitions. *Physical Review B*, 99(9):094205, Mar 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.094205. (Cited on page 46.)

[Dupont *et al.* 2019] Maxime Dupont, Nicolas Macé, and Nicolas Laflorencie. From eigenstate to hamiltonian: Prospects for ergodicity and localization. *Phys. Rev. B*, 100:134201, Oct 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.100.134201. (Cited on page 47.)

[Durr & Chakravarty 2019] Steven Durr and Sudip Chakravarty. Unsupervised learning eigenstate phases of matter. *Physical Review B*, 100(7):075102, Aug 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.100.075102. (Cited on pages 47 and 145.)

[Ebeling & Pöschel 1994] W. Ebeling and T. Pöschel. Entropy and long-range correlations in literary english. *Europhysics Letters (EPL)*, 26(4):241–246, May 1994. (Cited on pages 7 and 141.)

[Edmonds 1965] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. URL: https://www.cambridge.org/core/journals/canadian-journal-of-mathematics/article/paths-trees-and-flowers/08B492B72322C4130AE800C0610E0E21. (Cited on pages 110 and 111.)

[Fabiani & Mentink 2019] Giammarco Fabiani and Johan Mentink. Investigating ultrafast quantum magnetism with machine learning. *SciPost Physics*, 7(1):004, Jul 2019. URL: https://scipost.org/10.21468/SciPostPhys.7.1.004. (Cited on pages 80 and 85.)

[Ferrari *et al.* 2019] Francesco Ferrari, Federico Becca, and Juan Carrasquilla. Neural gutzwiller-projected variational wave functions. *Physical Review B*, 100(12):125131, Sep 2019. arXiv:1906.00463. (Cited on page 85.)

[Ferris & Poulin 2014] Andrew J. Ferris and David Poulin. Tensor networks and quantum error correction. *Physical Review Letters*, 113(3):030501, Jul 2014. arXiv:1312.4578. (Cited on page 112.)

[Feynman 1955] R. P. Feynman. Slow electrons in a polar crystal. *Phys. Rev.*, 97:660–665, Feb 1955. URL: https://link.aps.org/doi/10.1103/PhysRev.97.660. (Cited on page 5.)

[Figueirido *et al.* 1990] F. Figueirido, A. Karlhede, S. Kivelson, S. Sondhi, M. Rocek, and D. S. Rokhsar. Exact diagonalization of finite frustrated spin-(1/2 heisenberg models. *Physical Review B*, 41(7):4619–4632, Mar 1990. URL: https://link.aps.org/doi/10.1103/PhysRevB.41.4619. (Cited on page 82.)

[Fitzek *et al.* 2020] David Fitzek, Mattias Eliasson, Anton Frisk Kockum, and Mats Granath. Deep q-learning decoder for depolarizing noise on the toric code. *Phys. Rev. Research*, 2:023230, May 2020. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.023230. (Cited on pages 112, 113, 117, 119, 120 and 127.)

[Fowler 2015] Austin G. Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average o(1) parallel time. *Quantum Information and Computation*, 15(1–2):145–158, Jan 2015. (Cited on page 111.)

[Friedenauer *et al.* 2008] A. Friedenauer, H. Schmitz, J. T. Glueckert, D. Porras, and T. Schaetz. Simulating a quantum magnet with trapped ions. *Nature Physics*, 4(1010):757–761, Oct 2008. URL: https://www.nature.com/articles/nphys1032. (Cited on page 6.)

[Ganin *et al.* 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial Training of Neural Networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016. URL: http://dl.acm.org/citation.cfm?id=2946645.2946704. (Cited on page 67.)

[Gao & Duan 2017] Xun Gao and Lu-Ming Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8(11):1–6, Sep 2017. URL: https://www.nature.com/articles/s41467-017-00705-2. (Cited on pages 80, 126 and 154.)

[Garrison & Grover 2018] James R. Garrison and Tarun Grover. Does a single eigenstate encode the full hamiltonian? *Physical Review X*, 8(2):021026, Apr 2018. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.021026. (Cited on page 47.)

[Ge & Eisert 2015] Yimin Ge and Jens Eisert. Area laws and efficient descriptions of quantum many-body states, 2015. arXiv:1411.2995. (Cited on page 79.)

[Geiger *et al.* 2020] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, Feb 2020. arXiv:1901.01608. (Cited on pages 28 and 127.)

[Glasser *et al.* 2018] Ivan Glasser, Nicola Pancotti, Moritz August, Ivan D. Rodriguez, and J. Ignacio Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Physical Review X*, 8(1):011006, Jan 2018. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.011006. (Cited on page 79.)

[Glorot *et al.* 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. URL: http://proceedings.mlr.press/v15/glorot11a.html. (Cited on page 22.)

[Goodfellow *et al.* 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014)*, page 2672–2680, 2014. (Cited on page 16.)

[Goodfellow *et al.* 2015] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. arXiv:1412.6572. (Cited on pages 7, 125 and 141.)

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. (Cited on pages 22, 24, 26, 28 and 143.)

[Goodman & Flaxman 2017] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct 2017. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2741. (Cited on page 3.)

[Goremykina *et al.* 2019] Anna Goremykina, Romain Vasseur, and Maksym Serbyn. Analytically solvable renormalization group for the many-body localization transition. *Physical Review Letters*, 122(4):040601, Jan 2019. arXiv:1807.04285. (Cited on page 46.)

[Gornyi *et al.* 2005] I. V. Gornyi, A. D. Mirlin, and D. G. Polyakov. Interacting electrons in disordered wires: Anderson localization and low-temperature transport. *Physical Review Letters*, 95(20):206603, Nov 2005. arXiv:0506411. (Cited on page 43.)

[Gottesman 2009] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation, 2009. arXiv:0904.2557. (Cited on page 108.)

[Gottlieb von Windisch's 1784] Karl Gottlieb von Windisch's. *Inanimate reason; or a circumstantial account of that astonishing piece of mechanism, M. de Kemplen's Chefs-Player*. Bladon, 1784. (Cited on page 1.)

[Greitemann *et al.* 2019] Jonas Greitemann, Ke Liu, and Lode Pollet. Probing hidden spin order with interpretable machine learning. *Physical Review B*, 99(6):060404, Feb 2019. arXiv:1804.08557. (Cited on pages 42 and 126.)

[Greplova *et al.* 2017] Eliska Greplova, Christian Kraglund Andersen, and Klaus Mølmer. Quantum parameter estimation with a neural network, 2017. arXiv:1711.05238. (Cited on pages 7 and 141.)

[Gutzwiller 1963] Martin C. Gutzwiller. Effect of correlation on the ferromagnetism of transition metals. *Physical Review Letters*, 10(5):159–162, Mar 1963. URL: https://link.aps.org/doi/10.1103/PhysRevLett.10.159. (Cited on page 81.)

[Guyon *et al.* 1993] I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large vc-dimension classifiers. In *Advances in Neural Information Processing Systems*, page 147–155. Morgan Kaufmann, 1993. (Cited on page 22.)

[Hartnett 2019] Kevin Hartnett. Foundations built for a general theory of neural networks, 2019. URL: https://www.quantamagazine.org/foundations-built-for-a-general-theory-of-neural-networks-20190131/. (Cited on page 3.)

[Hastings *et al.* 2010] Matthew B. Hastings, Iván González, Ann B. Kallin, and Roger G. Melko. Measuring renyi entanglement entropy in quantum monte carlo simulations. *Phys. Rev. Lett.*, 104:157201, Apr 2010. URL: https://link.aps.org/doi/10.1103/PhysRevLett.104.157201. (Cited on pages 105 and 155.)

[Hastings 2007] M. B. Hastings. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(08):P08024–P08024, Aug 2007. (Cited on page 5.)

[He *et al.* 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 770–778, Jun 2016. doi:10.1109/CVPR.2016.90. (Cited on page 15.)

[Herbut 2007] Igor Herbut. *A Modern Approach to Critical Phenomena*. Cambridge University Press, 2007. doi:10.1017/CBO9780511755521. (Cited on page 36.)

[Hibat-Allah *et al.* 2020] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, Jun 2020. arXiv:2002.02973. (Cited on page 85.)

[Hinton & Salakhutdinov 2006] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, Jul 2006. URL: https://science.sciencemag.org/content/313/5786/504. (Cited on page 16.)

[Hinton *et al.* 2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, Jul 2006. URL: http://www.mitpressjournals.org/doi/10.1162/neco.2006.18.7.1527. (Cited on page 2.)

[Hsieh & LeGall 2011] Min-Hsiu Hsieh and François LeGall. Np-hardness of decoding quantum error-correction codes. *Physical Review A*, 83(5):052331, May 2011. URL: https://link.aps.org/doi/10.1103/PhysRevA.83.052331. (Cited on page 110.)

[Hsu *et al.* 2018] Yi-Ting Hsu, Xiao Li, Dong-Ling Deng, and S. Das Sarma. Machine learning many-body localization: Search for the elusive nonergodic metal. *Physical Review Letters*, 121(24):245701, Dec 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.245701. (Cited on pages 46, 47, 144 and 145.)

[Huang & Wang 2017] Li Huang and Lei Wang. Accelerated monte carlo simulations with restricted boltzmann machines. *Physical Review B*, 95(3):035105, Jan 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.035105. (Cited on pages 7 and 141.)

[Hubbard & Flowers 1963] J. Hubbard and Brian Hilton Flowers. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 276(1365):238–257, Nov 1963. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.1963.0204. (Cited on pages 3 and 139.)

[Huembeli *et al.* 2018] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. Identifying quantum phase transitions with adversarial neural networks. *Physical Review B*, 97(13):134109, Apr 2018. arXiv:1710.08382. (Cited on pages 43, 67 and 70.)

[Huembeli *et al.* 2019] Patrick Huembeli, Alexandre Dauphin, Peter Wittek, and Christian Gogolin. Automated discovery of characteristic features of phase transitions in many-body localization. *Physical Review B*, 99(10):104106, Mar 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.104106. (Cited on pages 47, 57, 67, 69 and 145.)

[Inack *et al.* 2018] E. M. Inack, G. E. Santoro, L. Dell'Anna, and S. Pilati. Projective quantum monte carlo simulations guided by unrestricted neural network states. *Physical Review B*, 98(23):235145, Dec 2018. arXiv:1809.03562. (Cited on page 85.)

[Ioffe & Szegedy 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL: http://proceedings.mlr.press/v37/ioffe15.html. (Cited on page 28.)

[Iso *et al.* 2018] Satoshi Iso, Shotaro Shiba, and Sumito Yokoo. Scale-invariant feature extraction of neural network and renormalization group flow. *Phys. Rev. E*, 97:053304, May 2018. URL: https://link.aps.org/doi/10.1103/PhysRevE.97.053304. (Cited on page 126.)

[Iten *et al.* 2020] Raban Iten, Tony Metger, Henrik Wilming, Lidia del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1):010508, Jan 2020. arXiv:1807.10300. (Cited on page 42.)

[Jastrow 1955] Robert Jastrow. Many-body problem with strong forces. *Physical Review*, 98(5):1479–1484, Jun 1955. URL: https://link.aps.org/doi/10.1103/PhysRev.98.1479. (Cited on page 81.)

[Jouppi *et al.* 2017] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul

Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, page 1–12, New York, NY, USA, 2017. Association for Computing Machinery. URL: https://doi.org/10.1145/3079856.3080246. (Cited on pages 2 and 78.)

[Kausar *et al.* 2020] Rubah Kausar, Wen-Jia Rao, and Xin Wan. Learning what a machine learns in a many-body localization transition. *Journal of Physics: Condensed Matter*, 32(41):415605, Jul 2020. URL: http://dx.doi.org/10.1088/1361-648X/ab9f09. (Cited on pages 47 and 145.)

[Khemani & Nandkishore 2020] Vedika Khemani and Rahul Nandkishore. Local constraints can globally shatter hilbert space: a new route to quantum information protection. *Physical Review B*, 101(17):174204, May 2020. arXiv:1904.04815. (Cited on page 92.)

[Khemani *et al.* 2017] Vedika Khemani, D. N. Sheng, and David A. Huse. Two universality classes for the many-body localization transition. *Phys. Rev. Lett.*, 119:075702, Aug 2017. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.075702. (Cited on pages 43, 46, 71, 76 and 150.)

[Kingma & Ba 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. arXiv:1412.6980. (Cited on pages 27 and 84.)

[Kitaev 2003] A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, Jan 2003. URL: http://www.sciencedirect.com/science/article/pii/S0003491602000180. (Cited on pages 8, 108 and 158.)

[Kjäll *et al.* 2014] Jonas A. Kjäll, Jens H. Bardarson, and Frank Pollmann. Many-body localization in a disordered quantum ising chain. *Physical Review Letters*, 113(10):107204, Sep 2014. URL: https://link.aps.org/doi/10.1103/PhysRevLett.113.107204. (Cited on page 49.)

[Koch-Janusz & Ringel 2018] Maciej Koch-Janusz and Zohar Ringel. Mutual information, neural networks and the renormalization group. *Nature Physics*, 14(6):578–582, June 2018. URL: https://doi.org/10.1038/s41567-018-0081-4. (Cited on page 126.)

[Kotov *et al.* 1999] Valeri N. Kotov, J. Oitmaa, Oleg P. Sushkov, and Zheng Weihong. Low-energy singlet and triplet excitations in the spin-liquid phase of the two-dimensional $J\_1{-}j\_2$ model. *Physical Review B*, 60(21):14613–14616, Dec 1999. URL: https://link.aps.org/doi/10.1103/PhysRevB.60.14613. (Cited on page 82.)

[Kramer 1991] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209. (Cited on page 30.)

[Krastanov & Jiang 2017] Stefan Krastanov and Liang Jiang. Deep neural network probabilistic decoder for stabilizer codes. *Scientific Reports*, 7(1):1–7, Sep 2017. URL: https://www.nature.com/articles/s41598-017-11266-1. (Cited on page 112.)

[Krizhevsky *et al.* 2017a] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. URL: https://dl.acm.org/doi/10.1145/3065386. (Cited on page 2.)

[Krizhevsky *et al.* 2017b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. URL: https://dl.acm.org/doi/10.1145/3065386. (Cited on pages 15, 24 and 25.)

[Laflorencie *et al.* 2020] Nicolas Laflorencie, Gabriel Lemarié, and Nicolas Macé. Chain breaking and kosterlitz-thouless scaling at the many-body localization transition, 2020. arXiv:2004.02861. (Cited on pages 49 and 126.)

[Lai *et al.* 2013] Hsin-Hua Lai, Kun Yang, and N. E. Bonesteel. Violation of the entanglement area law in bosonic systems with bose surfaces: Possible application to bose metals. *Phys. Rev. Lett.*, 111:210402, Nov 2013. URL: https://link.aps.org/doi/10.1103/PhysRevLett.111.210402. (Cited on pages 105 and 155.)

[Laughlin 1983] R. B. Laughlin. Anomalous quantum hall effect: An incompressible quantum fluid with fractionally charged excitations. *Phys. Rev. Lett.*, 50:1395–1398, May 1983. URL: https://link.aps.org/doi/10.1103/PhysRevLett.50.1395. (Cited on pages 3 and 139.)

[LeCun *et al.* 1989] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. (Cited on page 24.)

[LeCun *et al.* 1998] Yann LeCun, Corinna Cortes, and Christopher Burges. The mnist handwritten digit database, 1998. URL: http://yann.lecun.com/exdb/mnist/. (Cited on page 13.)

[LeCun *et al.* 2010] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, page 253–256. IEEE, May 2010. URL: http://ieeexplore.ieee.org/document/5537907/. (Cited on page 24.)

[LeCun *et al.* 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(75537553):436–444, May 2015. URL: https://www.nature.com/articles/nature14539. (Cited on page 2.)

[Lenggenhager *et al.* 2020] Patrick M. Lenggenhager, Doruk Efe Gökmen, Zohar Ringel, Sebastian D. Huber, and Maciej Koch-Janusz. Optimal renormalization group transformation from information theory. *Phys. Rev. X*, 10:011037, Feb 2020. URL: https://link.aps.org/doi/10.1103/PhysRevX.10.011037. (Cited on page 126.)

[Levine *et al.* 2019] Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. Quantum entanglement in deep learning architectures. *Physical Review Letters*, 122(6):065301, Feb 2019. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.065301. (Cited on page 80.)

[Li & Wang 2018] Shuo-Hui Li and Lei Wang. Neural network renormalization group. *Phys. Rev. Lett.*, 121:260601, Dec 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.260601. (Cited on page 126.)

[Li *et al.* 2015] Zi-Xiang Li, Yi-Fan Jiang, and Hong Yao. Solving the fermion sign problem in quantum Monte Carlo simulations by Majorana representation. *Physical Review B*, 91(24):241117, June 2015. URL: https://link.aps.org/doi/10.1103/PhysRevB.91.241117. (Cited on page 6.)

[Li *et al.* 2019] Zhenyu Li, Mingxing Luo, and Xin Wan. Extracting critical exponents by finite-size scaling with convolutional neural networks. *Physical Review B*, 99(7):075418, Feb 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.075418. (Cited on page 58.)

[Liu & Poulin 2019] Ye-Hua Liu and David Poulin. Neural belief-propagation decoders for quantum error-correcting codes. *Physical Review Letters*, 122(20):200501, May 2019. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.200501. (Cited on page 112.)

[Liu *et al.* 2017a] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning monte carlo method. *Physical Review B*, 95(4):041101, Jan 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.041101. (Cited on pages 7 and 141.)

[Liu *et al.* 2017b] Yue Liu, Tianlu Zhao, Wangwei Ju, and Siqi Shi. Materials discovery and design using machine learning. *Journal of Materiomics*, 3(3):159–177, Sep 2017. URL: http://www.sciencedirect.com/science/article/pii/S2352847817300515. (Cited on page 127.)

[Liu *et al.* 2018] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, page 19–34, 2018. (Cited on page 113.)

[Liu *et al.* 2019] Ke Liu, Jonas Greitemann, and Lode Pollet. Learning multiple order parameters with interpretable machines. *Physical Review B*, 99(10):104410, Mar 2019. arXiv:1810.05538. (Cited on page 126.)

[Luca & Scardicchio 2013] A. De Luca and A. Scardicchio. Ergodicity breaking in a model showing many-body localization. *EPL (Europhysics Letters)*, 101(3):37003, Feb 2013. (Cited on page 49.)

[Lucic *et al.* 2018] Mario Lucic, Karol Kurach, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Are gans created equal? a large-scale study. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 698–707, Red Hook, NY, USA, 2018. Curran Associates Inc. (Cited on page 68.)

[Luitz *et al.* 2015] David J. Luitz, Nicolas Laflorencie, and Fabien Alet. Many-body localization edge in the random-field heisenberg chain. *Physical Review B*, 91(8):081103, Feb 2015. arXiv:1411.0660. (Cited on pages 45, 49, 57, 59, 63, 73, 76, 147 and 148.)

[Macé & Alet 2019] Nicolas Macé and Fabien Alet. Numerical simulations of lattice models of many-body localization. unpublished, August 2019. (Cited on page 45.)

[Macé *et al.* 2019] Nicolas Macé, Fabien Alet, and Nicolas Laflorencie. Multifractal scalings across the many-body localization transition. *Physical Review Letters*, 123(18):180601, Oct 2019. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.180601. (Cited on pages 49, 62 and 67.)

[MacKay & Neal 1996] D.J.C. MacKay and R.M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645, 1996. URL: https://digital-library.theiet.org/content/journals/10.1049/el_19961141. (Cited on page 108.)

[MacQueen 1967] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL: https://projecteuclid.org/euclid.bsmsp/1200512992. (Cited on page 16.)

[Maskara *et al.* 2019] Nishad Maskara, Aleksander Kubica, and Tomas Jochym-O'Connor. Advantages of versatile neural-network decoding for topological codes. *Physical Review A*, 99(5):052351, May 2019. arXiv:1802.08680. (Cited on page 112.)

[Mehta & Schwab 2014] Pankaj Mehta and David J. Schwab. An exact mapping between the variational renormalization group and deep learning. *CoRR*, abs/1410.3831, 2014. arXiv:1410.3831. (Cited on pages 7, 126 and 141.)

[Mehta *et al.* 2019] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, May 2019. URL: http://www.sciencedirect.com/science/article/pii/S0370157319300766. (Cited on pages 16, 21, 23, 24, 28, 29, 30 and 142.)

[Melko & Sandvik 2005] Roger G. Melko and Anders W. Sandvik. Stochastic series expansion algorithm for the s=1/2 xy model with four-site ring exchange. *Physical Review E*, 72(2):026702, Aug 2005. arXiv:0409112. (Cited on page 90.)

[Melko *et al.* 2004] R. G. Melko, A. W. Sandvik, and D. J. Scalapino. Two-dimensional quantum XY model with ring exchange and external field. *Physical Review B*, 69(10):100408, Mar 2004. URL: https://link.aps.org/doi/10.1103/PhysRevB.69.100408. (Cited on pages 89 and 152.)

[Mitchell 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill, 1997. (Cited on page 12.)

[Moessner & Sondhi 2001] R. Moessner and S. L. Sondhi. Resonating valence bond phase in the triangular lattice quantum dimer model. *Phys. Rev. Lett.*, 86:1881–1884, Feb 2001. URL: https://link.aps.org/doi/10.1103/PhysRevLett.86.1881. (Cited on page 76.)

[Morningstar & Huse 2019] Alan Morningstar and David A. Huse. Renormalization-group study of the many-body localization transition in one dimension. *Physical Review B*, 99(22):224205, Jun 2019. arXiv:1903.02001. (Cited on page 46.)

[Morningstar *et al.* 2020] Alan Morningstar, David A. Huse, and John Z. Imbrie. Many-body localization near the critical point. *Phys. Rev. B*, 102:125134, Sep 2020. URL: https://link.aps.org/doi/10.1103/PhysRevB.102.125134. (Cited on page 46.)

[Moudgalya *et al.* 2019] Sanjay Moudgalya, Abhinav Prem, Rahul Nandkishore, Nicolas Regnault, and B. Andrei Bernevig. Thermalization and its absence within krylov subspaces of a constrained hamiltonian, 2019. arXiv:1910.14048. (Cited on page 92.)

[Nandkishore & Huse 2015] Rahul Nandkishore and David A. Huse. Many-body localization and thermalization in quantum statistical mechanics. *Annual Review of Condensed Matter Physics*, 6(1):15–38, Mar 2015. URL: https://www.annualreviews.org/doi/10.1146/annurev-conmatphys-031214-014726. (Cited on page 44.)

[Newman & Barkema 1999] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Clarendon Press, Feb 1999. (Cited on page 38.)

[Ni 2020] Xiaotong Ni. Neural network decoders for large-distance 2d toric codes. *Quantum*, 4:310, Aug 2020. arXiv:1809.06640. (Cited on pages 112 and 117.)

[Nielsen 2015] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com. (Cited on page 27.)

[Niu *et al.* 2019] Murphy Yuezhen Niu, Sergio Boixo, Vadim N. Smelyanskiy, and Hartmut Neven. Universal quantum control through deep reinforcement learning. *npj Quantum Information*, 5(11):1–8, Apr 2019. URL: https://www.nature.com/articles/s41534-019-0141-3. (Cited on page 127.)

[Nomura *et al.* 2017] Yusuke Nomura, Andrew S. Darmawan, Youhei Yamaji, and Masatoshi Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Phys. Rev. B*, 96:205152, Nov 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.96.205152. (Cited on pages 85 and 126.)

[Oganesyan & Huse 2007] Vadim Oganesyan and David A. Huse. Localization of interacting fermions at high temperature. *Physical Review B*, 75(15):155111, Apr 2007. URL: https://link.aps.org/doi/10.1103/PhysRevB.75.155111. (Cited on pages 43, 49 and 72.)

[Pal & Huse 2010] Arijeet Pal and David A. Huse. Many-body localization phase transition. *Physical Review B*, 82(17):174411, Nov 2010. URL: https://link.aps.org/doi/10.1103/PhysRevB.82.174411. (Cited on pages 43, 49 and 76.)

[Panda *et al.* 2020] R. K. Panda, A. Scardicchio, M. Schulz, S. R. Taylor, and M. Znidaric. Can we study the many-body localisation transition? *EPL (Europhysics Letters)*, 128(6):67003, Feb 2020. (Cited on pages 43, 45, 46, 71 and 150.)

[Paramekanti *et al.* 2002] Arun Paramekanti, Leon Balents, and Matthew P. A. Fisher. Ring exchange, the exciton bose liquid, and bosonization in two dimensions. *Physical Review B*, 66(5):054526, Aug 2002. URL: https://link.aps.org/doi/10.1103/PhysRevB.66.054526. (Cited on pages 88, 89, 91, 152, 153 and 154.)

[Paszke *et al.* 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. (Cited on page 29.)

[Pearson 1901] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. URL: https://doi.org/10.1080/14786440109462720, arXiv:https://doi.org/10.1080/14786440109462720. (Cited on page 16.)

[Pierleoni & Ceperley 2005] Carlo Pierleoni and David M. Ceperley. Computational methods in coupled electron–ion monte carlo simulations. *ChemPhysChem*, 6(9):1872–1878, Sep 2005. URL: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/cphc.200400587. (Cited on page 88.)

[Pietracaprina & Alet 2020] Francesca Pietracaprina and Fabien Alet. Probing many-body localization in a disordered quantum dimer model on the honeycomb lattice, 2020. arXiv:2005.10233. (Cited on page 76.)

[Pietracaprina *et al.* 2018a] Francesca Pietracaprina, Nicolas Macé, David J. Luitz, and Fabien Alet. Shift-invert diagonalization of large many-body localizing spin chains. *SciPost Physics*, 5(5):045, November 2018. URL: https://scipost.org/SciPostPhys.5.5.045. (Cited on page 4.)

[Pietracaprina *et al.* 2018b] Francesca Pietracaprina, Nicolas Macé, David J. Luitz, and Fabien Alet. Shift-invert diagonalization of large many-body localizing spin chains. *SciPost Physics*, 5(5):045, Nov 2018. URL: https://scipost.org/10.21468/SciPostPhys.5.5.045. (Cited on pages 49 and 146.)

[Pilania *et al.* 2013] Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran, and Ramamurthy Ramprasad. Accelerating materials property predictions using machine learning. *Scientific Reports*, 3(11):2810, Sep 2013. URL: https://www.nature.com/articles/srep02810. (Cited on pages 7 and 141.)

[Pilati & Pieri 2020] S. Pilati and P. Pieri. Simulating disordered quantum ising chains via dense and sparse restricted boltzmann machines. *Physical Review E*, 101(6):063308, Jun 2020. URL: https://link.aps.org/doi/10.1103/PhysRevE.101.063308. (Cited on page 85.)

[Ponte & Melko 2017] Pedro Ponte and Roger G. Melko. Kernel methods for interpretable machine learning of order parameters. *Physical Review B*, 96(20):205146, November 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.96.205146. (Cited on page 64.)

[Potirniche *et al.* 2019] Ionut-Dragos Potirniche, Sumilan Banerjee, and Ehud Altman. Exploration of the stability of many-body localization in d¿1. *Phys. Rev. B*, 99:205149, May 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.205149. (Cited on page 71.)

[Potter *et al.* 2015] Andrew C. Potter, Romain Vasseur, and S. A. Parameswaran. Universal properties of many-body delocalization transitions. *Physical Review X*, 5(3):031033, Sep 2015. arXiv:1501.03501. (Cited on page 46.)

[Poulin *et al.* 2009] David Poulin, Jean-Pierre Tillich, and Harold Ollivier. Quantum serial turbo codes. *IEEE Transactions on Information Theory*, 55(6):2776–2798, Jun 2009. doi:10.1109/TIT.2009.2018339. (Cited on page 110.)

[Preskill 2018] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, Aug 2018. arXiv:1801.00862. (Cited on pages 78 and 125.)

[Qi & Ranard 2019] Xiao-Liang Qi and Daniel Ranard. Determining a local hamiltonian from a single eigenstate. *Quantum*, 3:159, Jul 2019. arXiv:1712.01850. (Cited on page 47.)

[Rao 2018] Wen-Jia Rao. Machine learning the many-body localization transition in random spin systems. *Journal of Physics: Condensed Matter*, 30(39):395902, Sep 2018. (Cited on pages 47 and 145.)

[Real *et al.* 2017] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2902–2911. JMLR.org, 2017. (Cited on page 113.)

[Real *et al.* 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, page 4780–4789, 2019. (Cited on page 113.)

[Reinsel *et al.* 2018] David Reinsel, John Gantz, and John Rydning. The digitization of the world from edge to core, 2018. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. (Cited on page 2.)

[Rem *et al.* 2019] Benno S. Rem, Niklas Käming, Matthias Tarnowski, Luca Asteria, Nick Fläschner, Christoph Becker, Klaus Sengstock, and Christof Weitenberg. Identifying quantum phase transitions using artificial neural networks on experimental data. *Nature Physics*, 15(99):917–920, Sep 2019. URL: https://www.nature.com/articles/s41567-019-0554-0. (Cited on page 43.)

[Roger 1983] Michel Roger. Nuclear magnetism in solid 3he. *Journal of Magnetism and Magnetic Materials*, 31–34:727–732, Feb 1983. URL: http://www.sciencedirect.com/science/article/pii/0304885383906583. (Cited on pages 89 and 152.)

[Rosenblatt 1957] Frank Rosenblatt. The perceptron—a perceiving and recognizing automaton. *Report 85-460-1*, 1957. (Cited on pages 21 and 141.)

[Rousseau *et al.* 2004] V. Rousseau, G. G. Batrouni, and R. T. Scalettar. Phase separation in the two-dimensional bosonic hubbard model with ring exchange. *Physical Review Letters*, 93(11):110404, Sep 2004. URL: https://link.aps.org/doi/10.1103/PhysRevLett.93.110404. (Cited on pages 89, 90 and 152.)

[Rousseau *et al.* 2005] V. G. Rousseau, R. T. Scalettar, and G. G. Batrouni. Ring exchange and phase separation in the two-dimensional boson hubbard model. *Physical Review B*, 72(5):054524, Aug 2005. arXiv:0505347. (Cited on pages 89, 90 and 152.)

[Ruderman 1994] Daniel L Ruderman. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994. URL: https://doi.org/10.1088/0954-898X_5_4_006, arXiv:https://doi.org/10.1088/0954-898X_5_4_006. (Cited on pages 7, 24 and 141.)

[Rumelhart *et al.* 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, Oct 1986. URL: http://adsabs.harvard.edu/abs/1986Natur.323..533R. (Cited on pages 22 and 27.)

[Rupp *et al.* 2012] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, 108:058301, Jan 2012. URL: https://link.aps.org/doi/10.1103/PhysRevLett.108.058301. (Cited on pages 7 and 141.)

[Russakovsky *et al.* 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi:10.1007/s11263-015-0816-y. (Cited on page 12.)

[Sachdev 2011] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press, 2 edition, 2011. URL: https://www.cambridge.org/core/books/quantum-phase-transitions/33C1C81500346005E54C1DE4223E5562. (Cited on pages 36 and 37.)

[Sala *et al.* 2020] Pablo Sala, Tibor Rakovszky, Ruben Verresen, Michael Knap, and Frank Pollmann. Ergodicity breaking arising from hilbert space fragmentation in dipole-conserving hamiltonians. *Physical Review X*, 10(1):011047, Feb 2020. URL: https://link.aps.org/doi/10.1103/PhysRevX.10.011047. (Cited on page 92.)

[Salimans *et al.* 2017] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. arXiv:1703.03864. (Cited on page 127.)

[Sandvik *et al.* 2002] A. W. Sandvik, S. Daul, R. R. P. Singh, and D. J. Scalapino. Striped phase in a quantum xy model with ring exchange. *Physical Review Letters*, 89(24):247201, Nov 2002. URL: https://link.aps.org/doi/10.1103/PhysRevLett.89.247201. (Cited on pages 89 and 152.)

[Sandvik *et al.* 2010] Anders W. Sandvik, Adolfo Avella, and Ferdinando Mancini. Computational Studies of Quantum Spin Systems. In *Computational Studies of Quantum Spin Systems*, pages 135–338, Vietri sul Mare, (Italy), 2010. URL: http://aip.scitation.org/doi/abs/10.1063/1.3518900. (Cited on page 4.)

[Saraceni *et al.* 2020] N. Saraceni, S. Cantori, and S. Pilati. Scalable neural networks for the efficient learning of disordered quantum systems. *Physical Review E*, 102(3), Sep 2020. URL: http://dx.doi.org/10.1103/PhysRevE.102.033301. (Cited on page 43.)

[Schindler *et al.* 2017] Frank Schindler, Nicolas Regnault, and Titus Neupert. Probing many-body localization with neural networks. *Physical Review B*, 95(24):245134, Jun 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.245134. (Cited on pages 42, 47, 73, 75 and 145.)

[Schirber 2016] Michael Schirber. Nobel prize—topological phases of matter. *Physics*, 9, Oct 2016. URL: https://physics.aps.org/articles/v9/116. (Cited on page 37.)

[Schmidt *et al.* 2019] Jonathan Schmidt, Mário R. G. Marques, Silvana Botti, and Miguel A. L. Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(11):1–36, Aug 2019. URL: https://www.nature.com/articles/s41524-019-0221-0. (Cited on page 127.)

[Schuld *et al.* 2015] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015. URL: https://doi.org/10.1080/00107514.2014.964942, arXiv:https://doi.org/10.1080/00107514.2014.964942. (Cited on page 127.)

[Serbyn *et al.* 2013] Maksym Serbyn, Z. Papić, and Dmitry A. Abanin. Local conservation laws and the structure of the many-body localized states. *Physical Review Letters*, 111(12):127201, Sep 2013. arXiv:1305.5554. (Cited on page 49.)

[Serbyn *et al.* 2016] Maksym Serbyn, Alexios A. Michailidis, Dmitry A. Abanin, and Z. Papić. Power-law entanglement spectrum in many-body localized phases. *Physical Review Letters*, 117(16):160601, Oct 2016. URL: https://link.aps.org/doi/10.1103/PhysRevLett.117.160601. (Cited on page 74.)

[Sharir *et al.* 2020] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical Review Letters*, 124(2):020503, Jan 2020. arXiv:1902.04057. (Cited on page 85.)

[Silver *et al.* 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and et al. Mastering the game of go with deep neural net-

works and tree search. *Nature*, 529(75877587):484–489, Jan 2016. URL: https://www.nature.com/articles/nature16961. (Cited on pages 1, 8, 31 and 127.)

[Silver *et al.* 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and et al. Mastering the game of go without human knowledge. *Nature*, 550(76767676):354–359, Oct 2017. URL: https://www.nature.com/articles/nature24270. (Cited on pages 1, 31 and 127.)

[Silver *et al.* 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, and et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, Dec 2018. URL: https://science.sciencemag.org/content/362/6419/1140. (Cited on pages 2, 31, 32 and 127.)

[Slater 1930] J. C. Slater. Atomic shielding constants. *Physical Review*, 36(1):57–64, Jul 1930. URL: https://link.aps.org/doi/10.1103/PhysRev.36.57. (Cited on page 81.)

[Sorella 1998] S. Sorella. Green function monte carlo with stochastic reconfiguration. *Physical Review Letters*, 80(20):4558–4561, May 1998. arXiv:9803107. (Cited on pages 27 and 84.)

[Spigler *et al.* 2019] Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under- to over-parametrization affects loss landscape and generalization. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001, Nov 2019. arXiv:1810.09665. (Cited on pages 28 and 127.)

[Srivastava *et al.* 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL: http://jmlr.org/papers/v15/srivastava14a.html. (Cited on page 28.)

[Stanley & Miikkulainen 2002] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, page 2002, 2002. (Cited on pages 113, 114, 116, 117, 127 and 158.)

[Stanley *et al.* 2009] Kenneth Stanley, David D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, Jan 2009. URL: https://stars.library.ucf.edu/facultybib2000/2178. (Cited on page 123.)

[Stanley *et al.* 2019] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, Jan 2019. URL: http://www.nature.com/articles/s42256-018-0006-z. (Cited on page 127.)

[Steane 1996] Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, Nov 1996. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.1996.0136. (Cited on page 108.)

[Stoudenmire & Schwab 2016] Edwin Stoudenmire and David J Schwab. *Supervised Learning with Tensor Networks*, page 4799–4807. Curran Associates, Inc., 2016. URL: http://papers.nips.cc/paper/6211-supervised-learning-with-tensor-networks.pdf. (Cited on pages 7 and 141.)

[Suchsland & Wessel 2018] Philippe Suchsland and Stefan Wessel. Parameter diagnostics of phases and phase transition learning by neural networks. *Physical Review B*, 97(17):174435, May 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.97.174435. (Cited on page 42.)

[Suntajs *et al.* 2020] J. Suntajs, J. Bonca, T. Prosen, and L. Vidmar. Quantum chaos challenges many-body localization, 2020. arXiv:1905.06345. (Cited on page 45.)

[Sutton & Barto 2018] Richard Sutton and Andrew G. Barto. Reinforcement learning, second edition — the mit press, 2018. URL: https://mitpress.mit.edu/books/reinforcement-learning-second-edition. (Cited on pages 33 and 113.)

[Sweke *et al.* 2018] Ryan Sweke, Markus S. Kesselring, Evert P. L. van Nieuwenburg, and Jens Eisert. Reinforcement learning decoders for fault-tolerant quantum computation, 2018. arXiv:1810.07207. (Cited on pages 8, 31, 112, 113, 120 and 127.)

[Syljuåsen & Sandvik 2002] Olav F. Syljuåsen and Anders W. Sandvik. Quantum Monte Carlo with directed loops. *Physical Review E*, 66(4):046701, October 2002. URL: https://link.aps.org/doi/10.1103/PhysRevE.66.046701. (Cited on page 6.)

[Tay & Motrunich 2010] Tiamhock Tay and Olexei I. Motrunich. Study of a hard-core boson model with ring-only interactions. *Physical Review Letters*, 105(18), Oct 2010. arXiv:1007.3334. (Cited on pages 89, 92, 93, 96, 98, 102, 126, 152, 153, 154 and 155.)

[Tay & Motrunich 2011] Tiamhock Tay and Olexei I. Motrunich. Possible realization of the exciton bose liquid phase in a hard-core boson model with ring-only exchange interactions. *Physical Review B*, 83(20), May 2011. arXiv:1011.0055. (Cited on pages 89, 90, 91, 93, 94, 96, 101, 102, 105, 126, 152, 153 and 154.)

[Théveniaut & Alet 2019] Hugo Théveniaut and Fabien Alet. Neural network setups for a precise detection of the many-body localization transition: Finite-size scaling and limitations. *Phys. Rev. B*, 100:224202, Dec 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.100.224202. (Cited on pages 9, 36, 38, 43, 47, 48, 50, 53, 55, 57, 64, 66, 67, 70, 71, 125 and 150.)

[Théveniaut *et al.* 2020] Hugo Théveniaut, Zhihao Lan, Gabriel Meyer, and Fabien Alet. Transition to a many-body localized regime in a two-dimensional disordered quantum dimer model. *Physical Review Research*, 2(3):033154, Jul 2020. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033154. (Cited on pages 9, 36, 38, 71, 72, 73, 75, 76 and 150.)

[Thiery *et al.* 2018] Thimothée Thiery, François Huveneers, Markus Müller, and Wojciech De Roeck. Many-body delocalization as a quantum avalanche. *Phys. Rev. Lett.*, 121:140601, Oct 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.140601. (Cited on page 55.)

[Thomson & Schiró 2018] S. J. Thomson and M. Schiró. Time evolution of many-body localized systems with the flow equation approach. *Phys. Rev. B*, 97:060201, Feb 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.97.060201. (Cited on page 71.)

[Torlai & Melko 2017] Giacomo Torlai and Roger G. Melko. Neural decoder for topological codes. *Physical Review Letters*, 119(3):030501, Jul 2017. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.030501. (Cited on pages 8, 112, 119, 127 and 160.)

[Torlai *et al.* 2018] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(55):447–450, May 2018. URL: https://www.nature.com/articles/s41567-018-0048-5. (Cited on pages 7, 78 and 141.)

[Troyer & Wiese 2005] Matthias Troyer and Uwe-Jens Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Physical Review Letters*, 94(17):170201, May 2005. arXiv:0408370. (Cited on page 6.)

[Tsui *et al.* 1982] D. C. Tsui, H. L. Stormer, and A. C. Gossard. Two-dimensional magnetotransport in the extreme quantum limit. *Phys. Rev. Lett.*, 48:1559–1562, May 1982. URL: https://link.aps.org/doi/10.1103/PhysRevLett.48.1559. (Cited on pages 3 and 139.)

[van Nieuwenburg *et al.* 2017] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435–439, Feb 2017. arXiv:1610.02048. (Cited on pages 7, 38, 41, 47, 67, 125, 143 and 145.)

[van Nieuwenburg *et al.* 2018] Evert van Nieuwenburg, Eyal Bairey, and Gil Refael. Learning phase transitions from dynamics. *Phys. Rev. B*, 98:060301, Aug 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.98.060301. (Cited on pages 38, 42, 47 and 145.)

[Vargas-Hernández *et al.* 2018] Rodrigo A. Vargas-Hernández, John Sous, Mona Berciu, and Roman V. Krems. Extrapolating quantum observables with machine learning: Inferring multiple phase transitions from properties of a single phase. *Phys. Rev. Lett.*, 121:255702, Dec 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.255702. (Cited on page 43.)

[Varsamopoulos *et al.* 2017] Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1):015004, Nov 2017. (Cited on pages 8 and 112.)

[Varsamopoulos *et al.* 2020a] Savvas Varsamopoulos, Koen Bertels, and Carmen G. Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, Feb 2020. arXiv:1811.12456. (Cited on page 112.)

[Varsamopoulos *et al.* 2020b] Savvas Varsamopoulos, Koen Bertels, and Carmen G. Almudever. Decoding surface code with a distributed neural network based decoder. *Quantum Machine Intelligence*, Feb 2020. arXiv:1901.10847. (Cited on page 112.)

[Venderley *et al.* 2018] Jordan Venderley, Vedika Khemani, and Eun-Ah Kim. Machine learning out-of-equilibrium phases of matter. *Physical Review Letters*, 120(25):257204, Jun 2018. URL: https://link.aps.org/doi/10.1103/PhysRevLett.120.257204. (Cited on pages 43, 46, 47, 75, 144 and 145.)

[Verstraete *et al.* 2008] F. Verstraete, J. I. Cirac, and V. Murg. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, Mar 2008. arXiv:0907.2796. (Cited on pages 5 and 79.)

[Vosk *et al.* 2015] Ronen Vosk, David A. Huse, and Ehud Altman. Theory of the many-body localization transition in one dimensional systems. *Physical Review X*, 5(3):031032, Sep 2015. arXiv:1412.3117. (Cited on page 46.)

[Wahl *et al.* 2019] Thorsten B. Wahl, Arijeet Pal, and Steven H. Simon. Signatures of the many-body localized regime in two dimensions. *Nature Physics*, 15(22):164–169, Feb 2019. URL: https://www.nature.com/articles/s41567-018-0339-x. (Cited on page 71.)

[Wang 2016] Lei Wang. Discovering phase transitions with unsupervised learning. *Physical Review B*, 94(19), Nov 2016. arXiv:1606.00318. (Cited on pages 7, 40, 125 and 143.)

[Wetzel 2017] Sebastian Johann Wetzel. Unsupervised learning of phase transitions: from principal component analysis to variational autoencoders. *Physical Review E*, 96(2):022140, Aug 2017. arXiv:1703.02435. (Cited on pages 7, 40, 125 and 143.)

[Wietek *et al.* 2017] Alexander Wietek, Michael Schuler, and Andreas M. Läuchli. Studying continuous symmetry breaking using energy level spectroscopy, 2017. arXiv:1704.08622. (Cited on pages 4 and 140.)

[Wikipedia contributors 2020] Wikipedia contributors. Autoencoder — Wikipedia, the free encyclopedia, 2020. [Online; accessed 2-October-2020]. URL: https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=980751921. (Cited on page 30.)

[Williams 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. URL: https://doi.org/10.1007/BF00992696. (Cited on pages 33 and 123.)

[Wilson 1975] Kenneth G. Wilson. The renormalization group: Critical phenomena and the kondo problem. *Reviews of Modern Physics*, 47(4):773–840, Oct 1975. URL: https://link.aps.org/doi/10.1103/RevModPhys.47.773. (Cited on page 37.)

[Wolpert & Macready 1997] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997. URL: http://ieeexplore.ieee.org/document/585893/. (Cited on page 13.)

[Xu *et al.* 2017] Xiao Yan Xu, Yang Qi, Junwei Liu, Liang Fu, and Zi Yang Meng. Self-learning determinantal quantum monte carlo method. *Physical Review B*, 96(4), Jul 2017. arXiv:1612.03804. (Cited on pages 7 and 141.)

[Yin *et al.* 2017] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing, 2017. arXiv:1702.01923. (Cited on page 25.)

[Yu *et al.* 2016] Xiongjie Yu, David J. Luitz, and Bryan K. Clark. Bimodal entanglement entropy distribution in the many-body localization transition. *Phys. Rev. B*, 94:184202, Nov 2016. URL: https://link.aps.org/doi/10.1103/PhysRevB.94.184202. (Cited on page 57.)

[Zeiler 2012] Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012. arXiv:1212.5701. (Cited on page 27.)

[Zhang & Kim 2017] Yi Zhang and Eun-Ah Kim. Quantum loop topography for machine learning. *Physical Review Letters*, 118(21), May 2017. arXiv:1611.01518. (Cited on pages 8 and 38.)

[Zhang *et al.* 2017] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=Sy8gdB9xx. (Cited on pages 3, 28, 52 and 125.)

[Zhang *et al.* 2019a] Wei Zhang, Lei Wang, and Ziqiang Wang. Interpretable machine learning study of the many-body localization transition in disordered quantum ising spin chains. *Physical Review B*, 99(5):054208, Feb 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.054208. (Cited on pages 47, 58, 62 and 145.)

[Zhang *et al.* 2019b] Yi Zhang, A. Mesaros, K. Fujita, S. D. Edkins, M. H. Hamidian, K. Ch'ng, H. Eisaki, S. Uchida, J. C. Séamus Davis, Ehsan Khatami, and et al. Machine learning in electronic-quantum-matter imaging experiments. *Nature*, 570(7762):484–490, Jun 2019. URL: http://www.nature.com/articles/s41586-019-1319-8. (Cited on page 42.)

[Zhitomirsky & Ueda 1996] M. E. Zhitomirsky and Kazuo Ueda. Valence-bond crystal phase of a frustrated spin-1/2 square-lattice antiferromagnet. *Physical Review B*, 54(13):9007–9010, Oct 1996. URL: https://link.aps.org/doi/10.1103/PhysRevB.54.9007. (Cited on page 82.)