PhD Thesis

Machine Learning for Quantum Physics and Quantum Physics for Machine Learning

Author:
Patrick Huembeli

Supervisor:
Dr. Antonio Acín

February 6, 2021

Acknowledgement

It is hard to decide to whom I am most grateful to that the last four years went so well and that this PhD comes to a, hopefully, successful end. But the three people that for sure were most important for my academic success are my supervisor Antonio Acín, Peter Wittek and Alexandre Dauphin. I am grateful, that Toni gave me the opportunity to do my PhD in his group and for all the support, trust and freedom he gave me during the last years. Peter probably was the most influential person that ever entered my professional life. From the beginning, he took me under his wing, introduced me to the world of (quantum) machine learning and gave me the opportunity to connect with researchers all over the world. I will always remember his positive can-do attitude and his sense of humor. And last but not least, Alex guided me and my research on a daily basis. He always had time for discussions, helped with new ideas and asked important questions that improved my results and lead to new projects.

Life is a collection of many big and small interactions and so is this thesis. I want to thank all my collaborators and the people that supported me during the last years. A big thanks goes to all former and current members of the QIT group. It was always nice to work to a place with such a positive atmosphere. Furthermore, I want to thank Samyo, Hara, Daniel, Jessica and Vindhya who always had time for a quick coffee in the afternoon to discuss private and work matters and with whom I became close friends over the last years.

I am as well very thankful for all the opportunities I had to travel abroad and visit other groups. A very special thanks to Roger Melko who hosted me for four months in Waterloo and to Xanadu and the Creative Destruction Lab who made it possible to visit Toronto for another five months. These were really important experiences to me which let me grow as a researcher.

ICFO is an excellent place to do a PhD and I am very grateful for all the support I got from ICFO members in IT, HR, Travel and many more. Thank you very much for all your hard work.

And finally, I dedicate a special thank-you to my girlfriend, Helena, who always held my back. Even through this pandemic she managed to make it a great experience to work from home.

Abstract

Research at the intersection of machine learning (ML) and quantum physics is a recent growing field due to the enormous expectations and the success of both fields. ML is arguably one of the most promising technologies that has and will continue to disrupt many aspects of our lives. The way we do research is almost certainly no exception and ML, with its unprecedented ability to find hidden patterns in data, will be assisting future scientific discoveries. Quantum physics on the other side, even though it is sometimes not entirely intuitive, is one of the most successful physical theories and we are on the verge of adopting some quantum technologies of the second quantum revolution in our daily life. Quantum many-body physics is a subfield of quantum physics where we study the collective behavior of particles or atoms and the emergence of phenomena that are due to this collective behavior, such as phases of matter. The study of phase transitions of these systems often requires some intuition of how we can quantify the order parameter of a phase. ML algorithms can imitate something similar to intuition by inferring knowledge from example data. They can, therefore, discover patterns that are invisible to the human eye which makes them excellent candidates to study phase transitions. At the same time, quantum devices are known to be able to perform some computational task exponentially faster than classical computers and they are able to produce data patterns that are hard to simulate on classical computers. Therefore, there is the hope that ML algorithms run on quantum devices show an advantage over their classical analog.

This thesis is devoted to study two different paths along the front lines of ML and quantum physics. On one side we study the use of neural networks (NN) to classify phases of mater in many-body quantum systems. On the other side, we study ML algorithms that run on quantum computers. The connection between ML for quantum physics and quantum physics for ML in this thesis is an emerging subfield in ML, the interpretability of

learning algorithms. A crucial ingredient in the study of phase transitions with NNs is a better understanding of the predictions of the NN, to eventually infer a model of the quantum system and interpretability can assist us in this endeavor. The interpretability method that we study analyzes the influence of the training points on a test prediction and it depends on the curvature of the NN loss landscape. This further inspired an in-depth study of the loss of quantum machine learning (QML) applications which we as well will discuss.

In this thesis we give answers to the questions of how we can leverage NNs to classify phases of matter and we use a method that allows to do domain adaptation to transfer the learned "intuition" from systems without noise onto systems with noise. To map the phase diagram of quantum many-body systems in a fully unsupervised manner, we study a method known from anomaly detection that allows us to reduce the human input to a minimum. We will as well use interpretability methods to study NNs that are trained to distinguish phases of matter to understand if the NNs are learning something similar to an order parameter and if their way of learning can be made more accessible to humans. And finally, inspired by the interpretability of classical NNs, we develop tools to study the loss landscapes of variational quantum circuits to identify possible differences between classical and quantum ML algorithms that might be leveraged for a quantum advantage.

Resumen

La investigación en la intersección del aprendizaje automático (machine learning, ML) y la física cuántica es una área en crecimiento reciente debido al éxito y las enormes expectativas de ambas áreas. ML es posiblemente una de las tecnologías más prometedoras que ha alterado y seguirá alterando muchos aspectos de nuestras vidas. Es casi seguro que la forma en que investigamos no es una excepción y el ML, con su capacidad sin precedentes para encontrar patrones ocultos en los datos ayudará a futuros descubrimientos científicos. La física cuántica, por otro lado, aunque a veces no es del todo intuitiva, es una de las teorías físicas más exitosas, y además estamos a punto de adoptar algunas tecnologías cuánticas en nuestra vida diaria. La física cuántica de los muchos cuerpos (many-body) es una subárea de la física cuántica donde estudiamos el comportamiento colectivo de partículas o átomos y la aparición de fenómenos que se deben a este comportamiento colectivo, como las fases de la materia. El estudio de las transiciones de fase de estos sistemas a menudo requiere cierta intuición de cómo podemos cuantificar el parámetro de orden de una fase. Los algoritmos de ML pueden imitar algo similar a la intuición al inferir conocimientos a partir de datos de ejemplo. Por lo tanto, pueden descubrir patrones que son invisibles para el ojo humano, lo que los convierte en excelentes candidatos para estudiar las transiciones de fase.

Al mismo tiempo, se sabe que los dispositivos cuánticos pueden realizar algunas tareas computacionales exponencialmente más rápido que los ordenadores clásicos y pueden producir patrones de datos que son difíciles de simular en los ordenadores clásicos. Por lo tanto, existe la esperanza de que los algoritmos ML que se ejecutan en dispositivos cuánticos muestren una ventaja sobre su analógico clásico.

Esta tesis está dedicada a estudiar dos caminos diferentes a lo largo de la vanguardia del ML y la física cuántica. Por un lado, estudiamos el uso de redes neuronales (neural network, NN) para clasificar las fases de la materia en sistemas cuánticos de muchos cuerpos. Por otro lado, estudiamos los algoritmos ML que se ejecutan en ordenadores cuánticos. La conexión entre ML para la física cuántica y la física cuántica para ML en esta tesis es un subárea emergente en ML: la interpretabilidad de los algoritmos de aprendizaje. Un ingrediente crucial en el estudio de las transiciones de fase con NN es una mejor comprensión de las predicciones de la NN, para eventualmente inferir un modelo del sistema cuántico. Así pues, la interpretabilidad de la NN puede ayudarnos en este esfuerzo.

El método de interpretabilidad que estudiamos en esta tesis analiza la influencia de los puntos de entrenamiento en la predicción de una prueba que depende de la curvatura del paisaje de pérdidas de la NN. Esta dependencia inspiró además un estudio en profundidad de la pérdida de aplicaciones de aprendizaje automático cuántico (quantum machine learning, QML) que también discutiremos.

En esta tesis damos respuesta a las preguntas de cómo podemos aprovechar las NN para clasificar las fases de la materia y utilizamos un método que permite hacer una adaptación de dominio para transferir la "intuición" aprendida de sistemas sin ruido a sistemas con ruido. Para mapear el diagrama de fase de los sistemas cuánticos de muchos cuerpos de una manera totalmente no supervisada, estudiamos un método conocido de detección de anomalías que nos permite reducir la entrada humana al mínimo. También usaremos métodos de interpretabilidad para estudiar las NN que están entrenadas para distinguir fases de la materia para comprender si las NN están aprendiendo algo similar a un parámetro de orden y si su forma de aprendizaje puede ser más accesible para los humanos. Y finalmente, inspirados por la interpretabilidad de las NN clásicas, desarrollamos herramientas para estudiar los paisajes de pérdida de los circuitos cuánticos variacionales para identificar posibles diferencias entre los algoritmos ML clásicos y cuánticos que podrían aprovecharse para obtener una ventaja cuántica.

Contents

Chapter 1

Introduction

1.1 State of the Art

In recent years the field of machine learning (ML) has shown enormous progress and ML algorithms have been applied successfully to many tasks that influence our life in multiple ways. The range of applications is almost unlimited and goes from email spam filtering, credit card fraud detections, to voice, face and written word recognition, and many others. ML is a subfield of artificial intelligence that focuses on the study of algorithms that can improve their performance in an automated fashion through experience, which is provided in the form of sample data. The success of ML soon started to draw attention from science and learning algorithms, like neural networks (NNs), have been harnessed to solve problems of quantum chemistry, material science, and biology [1, 2, 3, 4]. Physics is no exception and ML methods have also been explored by astrophysics, high-energy physics, condensed matter physics, and quantum computing [5, 6, 7, 8, 9].

At the same time, parallel to the recent development in using ML algorithms on quantum physics, a new field is emerging: quantum machine learning (QML). As shown in figure 1.1 we distinguish QML from classical ML solely by the hardware it is run on. In this definition QML refers to ML tasks run on a quantum computer independent on whether the data is of classical or quantum origin, depicted in red. In green we depict ML assisted quantum physics, the field of classical ML algorithms applied on quantum data e.g. for the analysis of phase transitions.

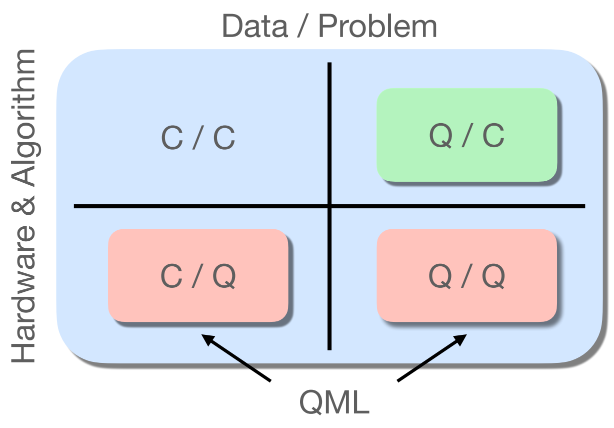


Figure 1.1: A schematic representation of (Q)ML tasks dependent on if the data or the hardware is classical or quantum (Q / C). We differentiate QML (in red) from ML assited quantum physics (in green) solely via the hardware that it is run on.

Phase Detection

Especially abundant is the use of ML for phase classification which is not surprising if one considers that determining the proper order parameters for unknown transitions is a highly non-trivial task. Quantifying states of matter is often guided by intuition of the researcher and requires a lot of experience and sometimes even educated guessing. An alternative route was shown, when NNs first located the phase transitions for known systems without a priori physical knowledge [10, 11]. These findings opened a new field of research following the question "Can ML offer a qualitative advantage by assisting scientific discovery in many-body physics? Or is it just a new tool for numerical calculations?"

The main goals of the use of ML to distinguish phases of matter are manyfold. First, it is not yet entirely clear what kind of phase transitions ML can detect and therefore, there are several studies that apply ML on all kind of classical [10, 12, 13, 14], quantum [11, 15, 16, 17, 18, 19], and topological phase transitions [20, 21, 22, 23]. Second, a key feature of ML applied to phase transition is its ability to classify states of matter without human input. This indicates that the task of finding an order parameter can be outsourced to the ML algorithm and we let it decide what features are relevant to distinguish the phases. Especially in phase transition like the many-body localization (MBL), where the standard Landau paradigm breaks down, it is not entirely clear how we can best delineate the MBL phase. Therefore, an automated extraction of relevant information is promising. Finally, ML algorithms are often opaque and the "reasoning" behind a certain prediction is not accessible by human beings. For the detection of phase transitions it is important for us, as researchers, to understand how a NN came to its conclusion that a certain state comes from one or another phase. To construct a model of the physical system and its phase transitions, e.g. to design an order parameter, we need to be able to extract this information from the NN. While this issue has been addressed with specially constructed ML architectures that allow interpretability [24, 19], it has so far not been possible to interpret more complex models for phase detection and there is a demand for tools that allow, as well, to interpret general models. To address this problem, we investigate new methods of how to leverage ML on data coming from quantum systems in the Chapters 3 and 4. We show how partially supervised and unsupervised methods can find phase transitions even for systems with added noise, for completely new, unknown phases and for the aforementioned MBL phase transition. Furthermore, we show that NNs also have some numerical advantage because they require less averaging for noisy systems. In Chapter 5 we use a ML interpretability method to better understand what a NN learns when it distinguishes phases of matter that is generally applicable and not restricted to special case uses. We will refer to these three topics as ML assisted quantum physics.

Quantum Variational Circuits

The discovery of the HHL algorithm [25] that allows to estimate the solutions of systems of linear equations on a quantum computer and its first experimental implementation [26] define a milestone for the field of QML. Even though the HHL algorithm is not an algorithm that is suited for noisy intermediate-scale quantum (NISQ) devices [27], it as well brought a lot of attention to near term quantum computers. HHL is strictly speaking not a QML algorithm, but many ML protocols rely on performing matrix and vector operations and therefore it was seen as an important step also for the QML community. The term QML is mostly used to refer to ML tasks that are run on a quantum computer with the expectation to gain some advantage over classical computation [28]. Classical ML shows its strength in finding patterns in data that are not visible to the human eye. Quantum computers on the other hand are known to produce output data patterns that are not simulateable on classical computers in reasonable time [29]. Hence, the idea to use quantum computers to run ML algorithms is driven by the hope that if a quantum computer can produce complex output patterns, it can also detect patterns that are difficult to recognize for a classical computer [30]. Recent results rigorously prove that there are quantum speedups for certain quantum kernel applications [31]. The discovery of some seminal quantum algorithms that allowed to translate classical ML problems on a quantum computer finally put the field of QML in the spotlight of recent research. These new algorithms, namely, the quantum principal component analysis (qPCA) [32] and the quantum support vector machine (qSVM) [33] used a hypothetical fully error corrected quantum computer to harness a quantum advantage for a data classification problem. Even before the appearance of the former algorithms, D-Wave showed that their quantum annealer can be used to perform a classical ML task called Boosting on their quantum annealer and named it QBoost [34]. In the last few years other companies, such as IBM and Google designed their own quantum computers that are, compared to D-Wave, universal which means they have a universal set of quantum gates that in theory allow to run any quantum computation. All of these devices are not error corrected and have of the order of 50 qubits and are therefore referred to as NISQ devices. These quantum computers do not allow to run algorithms like qPCA and qSVM because the quantum states would decohere before it would be possible to measure an outcome and gate errors would not allow to apply sufficiently many operations. To run ML applications on NISQ computer a common choice is the use of variational quantum circuits (VQCs) which are quantum algorithms composed of quantum gates that are parametrized by a set of free classical parameters. These VQCs have been applied on general optimization tasks such as for example in chemistry [35] or to find ground states of classical spin problem [36] and for ML problems such as classifying images [37]. Even though these applications show promising results and for some fault tolerant applications there is provable quantum advantage [38], it is not yet clear if there is indeed an advantage of using NISQ VQCs for ML tasks which is in contrast to the aforementioned qPCA that has a mathematical provable advantage over its classical analog [30]. It is of great interest to better understand NISQ era QML algorithms and to identify possible resources of a quantum advantage. Like for classical NNs, there are results that show that some VQC architectures have universal approximation properties [39, 40] that in theory allows to approximate any function on a compact set. Nevertheless, compared to classical ML literature there is a lack of general understanding of VQCs. There is no general understanding of the loss minimization and how the loss landscapes of QML algorithms look like. We do not understand yet how stable the minima of the loss landscapes are and how well they generalize to new data. Hence, a first step to better understand VQCs is the development of tools to study these issues.

1.2 Motivation and main contributions

This thesis focuses on three main questions. i) How can we efficiently leverage ML techniques, especially NN, to find phase transitions? ii) Can we interpret the decision process of the NN and gain more understanding about the phase transition? iii) And finally, can we use techniques from classical ML on quantum enhanced ML for a better understanding of VQCs? We study two main approaches to the first question (i) of how to identify phase transitions in quantum many-body systems with NNs and dedicate two separate Chapters 3 and 4 to these approaches. Chapter 5 is dedicated to question (ii) and the interpretability of NN predictions, where the NN is trained on phase transitions. Chapter 6 is about question (iii) and the study of the loss landscape of VQCs. In the following we introduce the main motivation and questions behind these research topics and our contribution to the field.

Phase detection with domain adaptation

The identification of phases of matter is a challenging task, especially in quantum mechanics. Determining the proper order parameters for unknown transitions is not trivial, on the verge of being an art. It includes the search in the exponentially large

Hilbert space and the examination of symmetries existing in the system, guided by intuition and educated guessing. Traditionally, physicists have to identify the relevant order parameters for the classification of the different phases. The findings in [10] that a simple NN can predict the phases of a classical Ising spin model triggered a whole new research area of many-body physics and ML which has produced spectacular successes in a short span of time [10, 12, 13, 14, 21, 22, 41, 16, 22, 18, 11, 19, 17, 23]. Deep fully connected and convolutional neural networks (CNN) have been applied to detect phase transitions in a variety of physical models, for classical [10, 12, 13, 14], quantum [11, 15, 16, 17, 18, 19], and topological phase transitions [20, 21, 22, 23] with supervised [10, 41, 21, 16, 22, 18] and unsupervised [42, 12, 11, 19, 17, 23] approaches as well as for experimental data [43, 44]. The main idea is to reinterpret a phase transition as a data driven process, which can be learned by a ML algorithm or in our case more concretely by a NN. If the phases are well understood, standard supervised deep learning can be used to find out from which phase a unknown test state comes [10, 17, 21]. Unsupervised techniques have so-far been used in classical systems [19, 45], rely on the knowledge that manually engineered features, such spin-spin correlators, capture the physics of the phase transition [16] or require several retrainings of the NN [11]. The problem we want to solve here, however, is qualitatively different. We want to automatically learn the unknown location of a phase boundary in a way, where we only need labeled data from deep inside a phase. And we want to achieve this without engineering features by hand and directly use the full wave function coming from exact diagonalization which remains a challenge for existing methods.

Our Contribution:

In Chapter 3, instead of defining and engineering an observable that indicates a phase transition, a so-called order parameter, we let the NN learn the relevant features to distinguish phases of matter from unprocesses raw state vector as input data. We use a technique called domain adversarial neural networks (DANN) that allows to extract invariant features from the input data. This is independent on where in the phase diagram the data comes from or if it comes from a noisy system. We show the success of this technique by applying it on two paradigmatic models: the

Su-Schrieffer-Heeger (SSH) model with disorder and the spin-1/2 Heisenberg chain in a random magnetic field that shows a many-body localization (MBL) phase transition. The method finds unknown transitions successfully and predicts transition points in close agreement with standard methods. In the case of the MBL phase transition it manages to do so with far less averaging over different realizations than standard order parameters. Furthermore, we show that the DANN shows better performance than standard transfer learning if the test points come from a Hamiltonian with added noise.

Reinterpreting a phase transition as a data anomaly

NNs have been shown to be excellent classifiers and they are also exceptionally good at generalizing their learned knowledge. Their main drawback is that in general classification settings they have to be trained in a supervised manner. Even the domain adversarial approach relies on two differently labeled sets of input states that come from different phases. Often there is a region in the phase diagram where one is sure to be in one or the other phase, e.g. in the limits of integrability of the models. But for a fully automated discovery of new phases this seems to be still a mayor restriction. Some unsupervised approaches to identify phase transitions show promising results [12, 11, 15, 19, 17, 23], but they all have their own drawbacks. Either they are hard to train and have to be retrained many times [11] or they so far only work for classical systems [19]. We leverage a method known from anomaly detection [46, 47, 48] that is based on NN autoencoders and allows us to fully unsupervised map out the phase diagram of the extended Bose-Hubbard model.

Our Contribution:

In Chapter 4 we demonstrate how to explore phase diagrams with automated and unsupervised ML to find regions of interest for possible new phases. In contrast to supervised learning, where data is classified using predetermined labels, we perform anomaly detection, where the task is to differentiate a normal data set, composed of one or several classes, from anomalous data. In this context, anomalous signifies that states coming from different phases than the states contained in the training set show qualitative differences. As a paradigmatic example, we explore the phase diagram of the extended Bose-Hubbard model in one dimension at exact integer filling. We employ deep NNs to determine the entire phase diagram in a completely unsupervised and automated fashion. As input data, instead of the wavefunction, we use scalable information, such as the the entanglement spectra and central tensors derived from tensor-networks algorithms for ground-state computation. Later we extend our method and use experimentally accessible data such as low-order correlation functions as inputs. Apart from mapping the whole phase diagram, our method allows us to reveal an unknown phase-separated region between supersolid and superfluid parts with unexpected properties. This region appears in the system in addition to the standard superfluid, Mott insulator, Haldane-insulating, and density wave phases.

Phase transitions and neural networks: Interpreting the black box

Despite their success, NNs normally do not allow any insight into the reasoning behind their predictions. The models that were first able to locate phase transitions for known systems without *a priori* physical knowledge [10, 11] are largely opaque. The same is true for the DANN and the anomaly detection with AE in the Chapters 3 and 4. ‘Artificial intelligence’ was provided by extracting it from data, which is in stark contrast with a physicists’ main driving force: the need to understand the underlying mechanisms of the process. The need for ML interpretability was already apparent after the first successful use of ML in physics. The numerical power of ML cannot be denied, but without interpretable predictions it might just be another numerical tool. At the moment, we rely on black-box predictions that solely return labels without any explanation or deeper motivation. Even in phase classification problems, where NNs have been often used, we usually cannot be fully confident that NNs learn order parameters. The necessity for interpretation has been already stressed by physicists, but proposed methods are either restricted to linear models [24], to particular NN architectures [19], or require pre-engineering of the data, which makes them very specific to both the ML and physical model [49].

Our Contribution:

In Chapter 5 we show how an interpretability method called influence function can be used in the classification of quantum phase transitions to understand what characteristics are learned by a ML algorithm. This method does not rely on the *a priori* knowledge on the order parameter or the system itself, and it is straightforwardly applicable to any physical model or experimental data with no dependence on the architecture of the ML model. An influence function is an approximation method of a leave one out (LOO) training that helps to determine which training examples are the most relevant for a given prediction. We demonstrate how influence functions can unravel the black box of NN when trained to predict the phases of the one-dimensional extended spinless Fermi-Hubbard model at half-filling. We discuss how to interpret the influence function and see if a relevant physical concept was indeed learned or if the prediction cannot be trusted. As well, we present that an interpretable NN can give additional information on the phase transitions, not provided to the algorithm explicitly.

Our results provide strong evidence that the NN correctly learns an order parameter describing the quantum transition of the Fermi-Hubbard model. Moreover, we demonstrate that influence functions not only allow checking that the network can predict new unknown phases, but even allow physicists to be guided in understanding patterns responsible for the phase transition.

VQC loss landscapes

In recent years, ML techniques enhanced by NISQ devices and especially variational quantum circuits (VQC) have attracted much interest and have already been benchmarked for certain problems. Inspired by classical deep learning, VQCs are trained by gradient descent methods which allow for efficient training over big parameter spaces. For NISQ sized circuits, such methods show good convergence. There are however still many open questions related to the convergence of the loss function and to the trainability of these circuits in situations of vanishing gradients. Furthermore, it is not clear how “good” the minima are in terms of generalization and stability against perturbations of the data and there is, therefore, a need for tools to quantitatively study the convergence of the VQCs. The use of the influence function inspired us to study the spectrum of the Hessian of a loss landscape that contains information of the curvature of the minima of classical NNs.

Our Contribution:

In Chapter 6 we introduce a way to study the loss landscape of VQCs with the spectrum of its Hessian. This opens a systematic way of quantifying loss landscapes and helps us to compare QML to classical ML algorithms. This idea of studying the Hessian was inspired by a ML interpretability method that we use in Chapter 5 and the interpretability of ML algorithms is the connecting piece between ML assisted quantum physic and QML in this thesis. We introduce a way to compute the Hessian of the loss function of VQCs and show how to characterize the loss landscape with it. The eigenvalues of the Hessian give information on the local curvature and we discuss how this information can be interpreted and compared to classical NNs. We benchmark our results on several examples, starting with a simple analytic toy model to provide some intuition about the behavior of the Hessian, then going to bigger circuits, and also train VQCs on data. Finally, we show how the Hessian can be used to escape flat regions of the loss landscape.

1.3 List of publications

This thesis is based on the following publications:

- *Identifying Quantum Phase Transitions with Adversarial Neural Networks*, P. Huembeli, A. Dauphin, P. Wittek , Physical Review B 97, 134109 (2018).

- *Automated discovery of characteristic features of phase transitions in many-body localization*, P. Huembeli, A. Dauphin, P. Wittek, C. Gogolin , Physical Review B 99, 104106 (2019).

- *Unsupervised phase discovery with deep anomaly detection*, K. Kottmann, P. Huembeli, M. Lewenstein, A. Acin, Physical Review Letter 125, 170603 (2020).

- *Phase Detection with Neural Networks: Interpreting the Black Box*, A. Dawid, P. Huembeli, Michal Tomza, M. Lewenstein, A. Dauphin, Accepted in New Journal of Physics (2020).

- *Characterizing the loss landscape of variational quantum circuits*P. Huembeli, A. Dauphin ArXiv preprint: 2008.02785.

Other works of the author that are not included in this thesis:

- *QuCumber: Wavefunction reconstruction with neural networks*, M. J. S. Beach, I. De Vlugt, A. Golubeva, P. Huembeli, R. G. Melko, E. Merali, G. Torlai , SciPost Phys. 7, 009 (2019).
- *Towards a heralded eigenstate preserving measurement of multi-qubit parity in circuit QED*, P. Huembeli, S. Nigg, Phys. Rev. A 96, 012313 (2017).

Chapter 2

Background

In this chapter we introduce some concepts and notions that will be used throughout this thesis. We review some basic concepts of ML, interpretability and VQCs. Furthermore, we discuss the notion of phase transitions and its mapping to a ML task. We will leave the detailed descriptions of the ML models, as well as the details about the physical many-body models that we will investigate, to the Chapters 3, 4 and 5. The details of about the VQCs is discussed in Chapter 6. Therefore, expert readers may skip this chapter.

2.1 Machine Learning: An overview

This section gives a brief general introduction into ML terminology and the methods that will be used for the phase transition prediction in the chapters 3, 4 and 5 in this thesis.

Supervised Learning

In supervised learning, the training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_i^N$ contains a feature vector $\boldsymbol{x}_i$ coming from an input space X together with a target value y_i coming from an output space Y. The training data is drawn from the unknown joint probability distribution P defined on $X \times Y$. The learning process aims to find a function $f : X \to Y$ that correctly predicts the target value $y_i \in Y$ of a given input $\boldsymbol{x}_i \in X$. The function f is parameterized with the parameters $\boldsymbol{\theta}$ and its output is the predicted target value $y_i' = f(\boldsymbol{x}_i, \boldsymbol{\theta})$. The aim of a supervised task is to tune the parameters $\boldsymbol{\theta}$ such that the model eventually is capable to predict the target value of new data samples from P that were not contained in the training data set. Most used examples for supervised learning are regression and classification tasks. In this thesis we only use classification and therefore we do not go into detail for regression problems. A classification is the task of assigning a discrete output (a class) to the input, where the inputs $\boldsymbol{x}_i$ can be anything from images, to text or in our case to detect phase transitions x_i can be a wavefunction or a physical observables of a many-body system. In classification tasks we refer to the target value y_i as the label of the input data $\boldsymbol{x}_i$. It contains the class information, for example if an image shows a cat or a dog or in our case it will say if an input wavefunction is a state coming from phase A or phase B of a system. To train a supervised model we define a loss $l(y_i, y_i')$ which quantifies the difference between the prediction of the model $y_i' = f(\boldsymbol{x}_i, \boldsymbol{\theta})$ and the actual label y_i and via gradient descent we minimize the empirical risk $\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x},y\in\mathcal{D}} l(f(\boldsymbol{x}, \boldsymbol{\theta}), y)$. Commonly used losses are the mean square error loss or the (binary) cross-entropy loss [50]. We say that a ML model generalizes well, if the generalization error $\sum_{(x,y)\tilde{P}} l(f(\boldsymbol{x}, \boldsymbol{\theta}), y)$ is small.

Transfer Learning and Domain Adaptation

Transfer learning refers to the generalisation of knowledge that was gained by the training of a model with one input data distribution and applying it to a new but similar data distribution or to the same data distribution but with a different tasks [51]. A task in this context is what the ML model is supposed to detect. For example one task could be to label states from different phases and another could be to detect entangled and states with no entanglement. Figure 2.1 shows the different areas of transfer learning and how it can be distinguished from the "usual" supervised learning setting.

Domain adaptation is a special kind of transfer learning, where the input space X and the output space Y of the ML model are the same but the distribution of the data $p(\boldsymbol{x}, y)$ for $\boldsymbol{x} \in X$ and $y \in Y$ changes [53]. A domain is the combination of the input space X, the output space Y and the associated probability distribution $p(\boldsymbol{x}, y)$. The training data set in this context comes from the so called source domain $\mathcal{S} = (X, Y, p_S)$. With the notation we indicate, that the source domain is defined by a source distribution p_S over $X \times Y$. Normally in supervised tasks the test set is coming from the same distribution as the training set

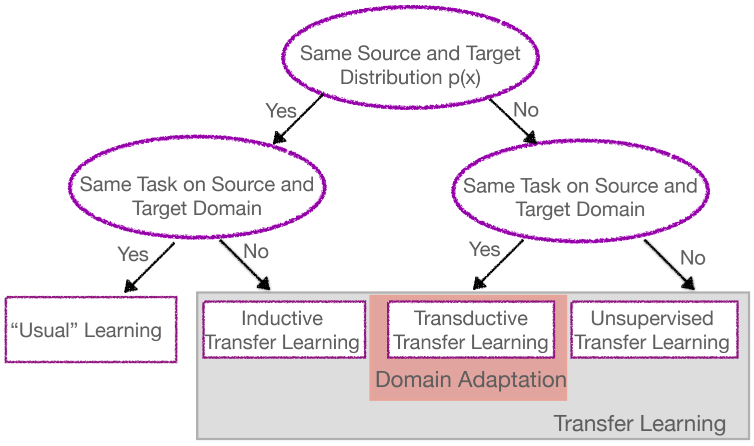

Same Source and Target Distribution p(x)
Yes
No
Same Task on Source and Target Domain
Yes
No
Same Task on Source and Target Domain
Yes
No
"Usual" Learning
Inductive Transfer Learning
Transductive Transfer Learning
Unsupervised Transfer Learning
Domain Adaptation
Transfer Learning

Figure 2.1: Comparing the usual supervised setting with transfer learning and positioning domain adaptation inside transfer learning via the distribution of the input data $p(\boldsymbol{x}, y)$ and the task that has to be fulfilled. Domain adaptation is the setting where the source and the target domain have different input data distributions, but for both domains the same task has to be learned. Figure is inspired by [52]

and therefore there is only a single domain. In domain adaptation tasks the test set stems from a different domain, which we call the target domain $\mathcal{T} = (X, Y, p_T)$. An example for domain adaptation is the MNIST-M dataset [54], which is the original "Modified National Institute of Standards and Technology" (MNIST) handwritten digits dataset with new color schemes (see Figure 2.2). The source set is the original MNIST data and the target set are MNIST images with colors. Already with such a simple setup, standard NN training schemes without domain adaptation struggle to identify the images with new colors, because the classifiers do not know what to do with the additional color information. Domain adaptation on the other side helps the NN to identify invariant features of both domains. In our case for the detection of phase transitions domain adaptation can be used for models with noise, where the source domain are the eigenstates of a system without noise and the target domain are the eigenstates of a system with noise.

The general idea of domain adaptation is that there is a domain invariant space, where the data of the source and the target domain can be represented indistinguishably and therefore the classification can be done independent on which domain the data

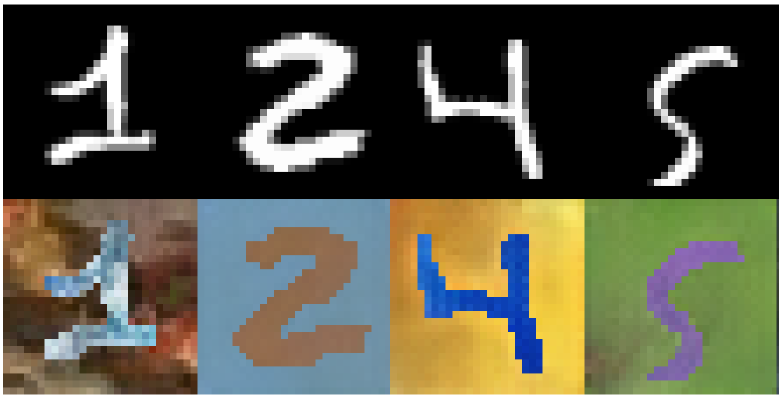


Figure 2.2: A sample from the original MNIST hand written digits dataset in black and white and the extended MNIST-M dataset with different color schemes for the digits and the backgrounds.

comes form. Domain adaptation focuses on finding this domain-invariant space.

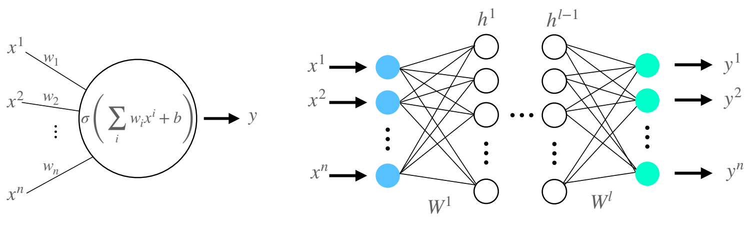
Neural Networks

NNs are a way to parametrize a function. The building blocks of a NN are artificial neurons which are functions that take an input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and output a value $y = \sigma(\sum_i w_i x_i + b)$ (See Figure 2.3(a)). The parameters w_i are called weights and b is called the bias of the neuron. We emphasize here that throughout the manuscript $\boldsymbol{x}_i$ refers to the whole input vector and x_i is a single element of the vector $\boldsymbol{x}_i$. The function $\sigma(\cdot)$ is an arbitrary non-linear function which we call activation function. Commonly used activation functions are the rectified linear units (ReLU) and the sigmoid function as described in Equation 2.1 and 2.2.

$$\text{ReLU: } f(x) = \max(0, x) \tag{2.1}$$

$$\text{Sigmoid: } f(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

A NN is a combination of several such neurons, where the outputs of some neurons are the inputs of others. To train a NN it is imperative to have a way to efficiently calculate its gradient. To do so, we organize its neurons in layers. A layer is a group of neurons that are not connected within each other but with the same input (See Figure 2.3(b)). The output of the i-th layer of neurons is only the input of the $(i+1)$-th layer and they are connected via wires with weights expressed as a matrix W^i with the single neuron weights w_i. If we structure the neurons in this manner, we are able to take the gradient layer by layer and use the chain rule for derivatives. This procedure is often referred to as backpropagation [55]. The first layer of a NN is normally referred to as the input layer where the data $\boldsymbol{x} \in X$ is used as an input. The last layer of a NN is referred to as the output layer of the NN and it returns the prediction of the label or regression y. The intermediate layers are referred to as hidden layers h^i. The weights and biases of all these layers are the parameters that have to be adjusted during the training. Since each neuron in one layer is connected to all the neurons in a consecutive layer, this kind of NN is also referred to as a fully connected NN. The notion of deep learning refers to the amount of layers used in a NN, but it is not clearly defined what amount of hidden layers classifies as deep. In this thesis we will refer to NNs that have more than one hidden layer as deep and NNs with only one hidden layer as shallow.



(a) (b)

Figure 2.3: Perceptron and Neural Network: a) A schematic representation of a neuron: A non-linear function $\sigma(\cdot)$ processes an input vector $\boldsymbol{x}$ by taking the dot product $\boldsymbol{w} \cdot \boldsymbol{x}$ with a weight vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ and by adding a bias value b. b) A NN is a combination of several neurons organized in layers.

Deep learning has been successfully used on countless problems in the last decade. From image recognition [56] to natural language processing [57] and to generate videos and music [58] that has never been played by humans. Cars learn how to drive autonomously and robots learn how to walk and grab objects thanks to NNs. They have been proven to be very versatile and they are also relatively easy to train via gradient based methods, such as stochastic gradient descent. Theoretical results show that NNs are universal approximators [59] which means that any continuous function $f : X \to Y$ on a compact input space X can be approximated by a NN with only one hidden layer. It also has been shown that the use of deep NNs allows us to learn certain problems with much less neurons compared to a shallow neural network [60], which hints at a possible explanation of why deep learning is so successful.

Convolutional Neural Network

Fully connected NNs have certain shortcomings. For example the number of weights scales with $\mathcal{O}(n_i \cdot n_{i+1})$ with the number of neurons n_i and n_{i+1} of two consecutive layers. For big input data, such as images of millions of pixels or in our case a quantum state vector of m spins and $\mathcal{O}(2^m)$ dimensions, the number of weights in fully connected NNs is growing fast already for shallow NNs and optimization becomes infeasible. If we want to be able to build deep architectures, we have to use less weights per layer. Furthermore, fully connected NNs, as well, do not take into account spatial hierarchies of features of the input data [61] because we for example have to reshape images to 1-dimensional vectors. This implies, for example for image data, that a fully connected NN does not take into account that pixels that are spatially close in an image should also show more correlation than pixels far apart in an image. The convolutional neural network (CNN) architecture takes this shortcoming into account and also helps to reduce the number of weights substantially. The main idea behind a CNN is depicted in Figure 5.4 and as its name says, a small so-called kernel matrix (often also referred to as a filter) is used as so-called receptive field that convolves through the input data. A kernel is nothing else than group of neurons that take spatially limited amount of input values from the previous layer and outputs a value for the next layer. In contrast to the fully connected NN this group of neurons is moved along the input data vector and does the same calculation for each position on the input vector with the same weights. Therefore one kernel convolves the whole input and the number of weights does not depend on the input size anymore but only on the number of input elements that are processed by the kernel, i.e. the size of the kernel. Later we will use one-dimensional and two-dimensional CNNs, dependent on if the input data is a vector or a matrix.

Pooling is another technique that is used in convolutional NNs to reduce the dimensionality of the representation between the NN layers, often referred to as a feature representation. A typical way of pooling is maximum pooling, which takes a certain amount of output values of a kernel and takes the maximum of them. Typically the pooling is also done with a receptive field that slides along the outputs of a convolutional layer and pools the values inside the receptive field together to a single value. Typical pooling functions are the mean and the maximum value of the inputs.

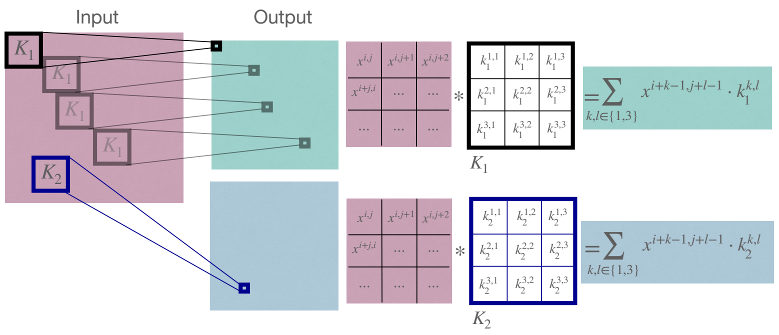

Input
Output
K_1
K_2
$x^{i,j}$ $x^{i,j+1}$ $x^{i,j+2}$
$x^{i+j,i}$ ... ...
... ...
$*$
$k_1^{1,1}$ $k_1^{1,2}$ $k_1^{1,3}$
$k_1^{2,1}$ $k_1^{2,2}$ $k_1^{2,3}$
$k_1^{3,1}$ $k_1^{3,2}$ $k_1^{3,3}$
$=\sum_{k,l\in\{1,3\}} x^{i+k-1,j+l-1}\cdot k_1^{k,l}$
$k_2^{1,1}$ $k_2^{1,2}$ $k_2^{1,3}$
$k_2^{2,1}$ $k_2^{2,2}$ $k_2^{2,3}$
$k_2^{3,1}$ $k_2^{3,2}$ $k_2^{3,3}$
$=\sum_{k,l\in\{1,3\}} x^{i+k-1,j+l-1}\cdot k_2^{k,l}$

Figure 2.4: Convolutional Neural Network: Schematic representation of a CNN in 2D with two different kernels K_1 and K_2. A Kernel of a given size, here with 3×3 pixels convolves along the input image and for each position $\{i, j\}$ on the input image a kernel K_m returns a value $\sum_{k,l\in\{1,3\}} x^{i+k-1,j+l-1} \cdot k_m^{k,l}$. Optionally, a kernel can also have an additional bias term and normally the kernel output is furthermore processed via a non-linear function $\sigma(\cdot)$.

Unsupervised Learning

In contrast to supervised learning, for unsupervised learning the data $\mathcal{D} = \{(\boldsymbol{x}_i)\}_i^N$ does not have a target value y_i and instead of learning an output value, the aim of supervised learning problems is to find (hidden) structures in data. Typical tasks for unsupervised learning include the clustering of data, where groups within a given dataset are discovered, the estimation of probability densities of the data and the learning of low-dimensional representations of data to better visualize it while maintaining the original structure of the data. Typical examples of the latter are the principal component analysis (PCA) and T-distributed Stochastic Neighbor Embedding (t-SNE) which we will later use in section 3 to post-process the latent variable from a NN.

NN Autoencoders

NN autoencoders are an example of an unsupervised ML method. An autoencoder consists of two consecutive NNs $f_{\boldsymbol{\theta}} : X \to Z$ and $g_{\boldsymbol{\phi}} : Z \to X$, parametrized by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. The function $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the encoding part of the NN that maps the input data $\boldsymbol{x} \in X$ to a, normally, lower dimensional space $z \in Z$. Throughout this thesis we will use the notation $f(\boldsymbol{x}, \boldsymbol{\theta})$. $\bar{\boldsymbol{x}} = g_{\boldsymbol{\phi}}(\boldsymbol{z}) = g(\boldsymbol{z}, \boldsymbol{\phi})$ is the decoder that maps the lower dimensional representation $\boldsymbol{z}$ back to $\boldsymbol{x'} \in X$. The aim of the training of a NN autoencoder is to find a low dimensional representation $\boldsymbol{z} \in Z$ that allows a decoding such that $\boldsymbol{x} \approx \bar{\boldsymbol{x}}$. Instead of comparing the NN output $\boldsymbol{x'}$ with a label like in a supervised NN setting, we compare it with the input $\boldsymbol{x}$. To do this we define a loss function $l(\bar{\boldsymbol{x}}, \boldsymbol{x}) = l(g(f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\phi}), \boldsymbol{x})$. The training itself can be done via gradient descent and the minimization of the average loss over the whole data set $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{\boldsymbol{x} \in \mathcal{D}} l(\bar{\boldsymbol{x}}, \boldsymbol{x})$. The mean square error is commonly used in this setting.

Machine Learning Interpretability

As stated in [62], the age of ML will bring an enormous reduction in the cost of prediction. Often confused with a general artificial intelligence, current ML algorithms are simple prediction machines. At least for supervised tasks. One one side they also have shown great success in learning from complex data and at predicting new patterns. The actual learning process on the other side is not accessible for humans and it is difficult for us to understand, why a ML algorithm came to a certain conclusion. In this section we give a brief introduction into the field of interpretability of ML, which is the study of why a ML model came to a certain prediction. We would like to outline a rough idea what interpretability means and emphasize that this itself is an ongoing research question [63] and therefore we only explain the basic idea of interpretability. In chapter 5 we apply an interpretability method called Influence Function, on the ML prediction of phases and explain it in more detail.

In [64] the authors point out that the need for interpretability arises from an incompleteness in problem formalization. For certain ML prediction tasks, it is not sufficient to get the prediction (the "what") with high accuracy. The model must also explain how it came to the prediction (the "why") because a correct prediction only partially solves the original problem. In physics, and in our particular case of the prediction of phase transitions, the "why" is of highest importance because we as scientists would like to gain new insight and understanding of the physics of a model via the NN predictions. The emphasis of interpretability is expressed in the following statements: "Interpretability is the degree to which a human can understand the cause of a decision" [65] or "the degree to which a human can consistently predict the model's result" [66] and "The ability to explain or to present in understandable terms to a human" [64]. For the task of predicting phase transitions with ML the motivation is not necessarily that a human can easily understand the underlying physics but the construction of new models is primarily in the focus. As stated in [67] "interpretable ML refers to methods and models that make the behavior and predictions of ML systems understandable to humans". To predict a phase transition we want to be capable of extracting information about the decision process that eventually lead to a model that helps us to understand the different phases.

The interpretability of ML algorithms is divided into two main categories. There are intrinsic interpretable and post-hoc interpretable models. Intrinsic interpretable models are self- explanatory and their interpretability is built directly into their structure. Examples for this kind of models are decision trees, rule based models, but also class activation maps [68]. Post-hoc interpretable models are black boxes and require additional tools and models to do interpretation. Post-hoc interpretability is also often referred to as model-agnostic interpretability [69] and is generally done after the training of a model [70]. In practice for post-hoc interpretability the focus lies mostly on the test points and through the perturbation of the features of the input data [71] one can understand how the fitted model behaves under these perturbations and how important a certain feature is. Another way is to locally fit a simpler model (e.g. a linear one) to a test point for Local Interpretable Model-agnostic Explanations (LIME) [72]. Therefore post-hoc analysis is primarily focused on the test dataset and not on the training of the model itself.

The influence function [73, 74, 75, 76, 77] that we use and introduce in full detail in Chapter 5 focuses more on the question of how the model arrived at this test point prediction and asks what would happen if a certain training point was excluded from the training. This is still in the spirit of the post-hoc analysis but with a slightly rephrased question and with less focus on the features of the test point but more on the training set as a whole.

2.2 What is a Phase Transition?

This section is dedicated to give some background in (quantum) phase transitions and clarify some terminology around many-body physics. The description of the specific physical models will follow in the respective chapters when we apply ML to identify the phase transitions. Concretely, in Chapter 3 we study the Su-Schrieffer-Heeger (SSH) model that hosts a non-trivial topological phase and the Heisenberg spin-1/2 model in a random magnetic field that shows a many-body localization (MBL) phase transition. In Chapter 4 we investigate the extended Bose-Hubbard model with its rich phase diagram and in Chapter 5 we use the extended Fermi-Hubbard model as a benchmark.

The study of phase transitions often requires a deep understanding and some intuition of the underlying physics. Especially, for more "exotic" phase transitions such as many-body-localization it is not yet entirely clear how to exactly identify the critical point as the underlying physics is not fully understood and standard Landau theory is not applicable because it is a phase outside thermodynamic equilibrium. Therefore, this section is not only an introduction into phase transitions themselves, but it will also show why the use of ML to identify phases of matter is promising. We start with a very general overview and refer to the later chapters for a more specific introduction to physical models and their phase transitions.

Most people get for the first time in contact with phases of matter and their transitions on the example of melting and vaporising ice, where the phase of water goes from solid to liquid and from liquid to gas. For everyday life it is often sufficient to know that there are these three states of matter. As we will see in this thesis there are many more of them. Furthermore, it is also worth mentioning that a phase transition is a very general phenomenon and does not necessarily refer to a state of matter but it can be a more abstract concept. Just to name two examples for such abstract transitions, a computational phase transition refers to a sudden change in satisfiability or run-time of a computational problem [78], and a Hopfield network experiences a phase transition if too many patterns are stored and the memory states become unstable [79].

The microscopic study of a many-body system like water comes with the challenge of the exploding numbers of degrees of freedom with growing system size if we would want to keep track of every single molecule. Thermodynamics on the other side allows us to describe a macroscopic system with a few variables such as temperature and pressure. In the case of melting ice, a smooth change of a thermodynamic parameter, e.g. the temperature, leads to an abrupt change in the state of matter which results in the transition from ice to water or from water to steam. This is called a phase transition and the temperature at which it occurs is called the critical temperature. Probably the most famous model for such a thermal phase transition in physics is the Ising model that changes its state with increasing temperature from an ordered (anti)ferromagnetic to a disordered paramagnetic state. The systems that we will study in this thesis are at zero temperature and their phase transitions are of pure quantum nature. The distinction between classical and quantum phase transition (CPT and QPT) is a distinction of non-zero and zero temperature transitions. Both, QPT and CPT can occur because of the change of the external conditions of the system such as temperature (for CPT only), pressure or magnetic field. As we later will see, QPTs also occur because of competing terms in the Hamiltonian of the system.

Thermal Fluctuation

The reason for the occurrence of a transition at non-zero temperature T is the competition between the inner energy and the entropy in the free energy $F = U - ST$ which is minimal if the system is in equilibrium. The inner energy U tends to be minimized in ordered phases and the entropy S is maximal for disordered states. With growing temperature T the contribution of the entropy on the free energy increases and the system will end up in a higher energy state U. For high temperatures the entropy term becomes dominant and the order that may exist gets destroyed.

This is often referred to as thermal fluctuations. More formally, according to the Ehrenfest classification of phases, one can describe a CPT as a singularity in the free energy or its derivative with respect to a thermodynamic variable. If the free energy or its derivative becomes singular also other thermodynamic quantities such as the entropy and the specific heat become singular.

To come back to our initial example, when transiting from one phase to another the melting of ice is among the examples that we will study in this thesis a special case, because the two phases water and ice can coexist. This is possible, because at melting temperature $T = 0°C$ there is some amount of latent heat needed to transform the remaining ice into water. If the phases become indistinguishable if we approach the critical point and one phase transits continuously into another phase we call them continuous or second order phase transitions. The notion of second order comes from the second derivative of the free energy in CPT which is discontinuous. As we will later see, a similar notation will also be used for Landau's theory of phase transition. And the order of the transition refers to the continuity of the derivatives of the order parameters. If the critical point is approached the spatial correlations become long ranged and diverge as $\chi \approx (|T - T_C|/T_C)^{-\nu}$, where ν is the critical exponent of the correlation length. With a diverging correlation length other observable quantities diverge as well. The collective behaviour of all these quantities at the critical point and their power law decay is usually referred to as critical behaviour.

Quantum Fluctuation

In the quantum case, instead of the entropy competing with the inner energy, competing parameters of the basic interactions of a system lead to phase transitions. Instead of thermal fluctuations, quantum fluctuations are responsible for the change of the phase. More formally this means that certain parts of the Hamiltonian compete with each other. To illustrate this on a simple example we introduce the transverse field Ising model in one dimension,

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - h \sum_i^N \sigma_i^x, \tag{2.3}$$

where J is the interaction strength between the spins, h is the transverse field strength, $\sigma^{\{x,z\}}$ are the pauli spin operators, $\langle i, j\rangle$ indicates the sum over nearest neighbours and N is the total number of spins. In this simple model, the interaction term leads to order, which indicates in the extreme case of $h = 0$ and $J > 0$ the lowest energy state can be achieved if the spins all align in the σ^z direction. An increasing transverse field h on the other side leads to spins aligning in the σ^x direction. The ground state of the transverse field part of the Hamiltonian is the product state $|+\rangle^N$ of the single spin equal superposition $|+\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$. Therefore, the ground state of the Hamiltonian is in a competition between its interaction and the field part.

Order Parameter

The Ehrenfest classification depends on a thermodynamical description of the system and its phases. Quantum phase transition occur at zero temperature and we, therefore, use a different way to study and quantify phases. A typical approach is to characterize them with a “degree of order” introduced by Landau [80], which nowadays referred to as an order parameter, that mostly is a local observable of the system. The function of an order parameter is to describe the order of a phase compared to another phase to determine its boundaries. It is non-zero for an ordered and zero for a disordered phase. This again can be a rather abstract notion of order and depends on the system. Furthermore, the order parameter captures the symmetries of the ordered phase and its values will go to zero if the symmetry is spontaneously broken at the phase transition. This definition holds for both classical and quantum systems. This so called Landau theory of phases can describe spontaneous symmetry breaking phases and the transition between these phases. We will also encounter quantum systems where spontaneous symmetry breaking will not be sufficient to characterize the phases of a system and no order parameter in the sense of the Landau theory exists. Examples for such "new" phases are topological phases or MBL.

A possible order parameter of our previous example, the transverse field Ising-model, is the average magnetization of the system $M = 1/N \sum_i^N \langle\sigma_i^z\rangle$ that is $\neq 0$ in the ordered phase. In the disordered phase, where $h \gg J$ and the magnetization $M = 0$, the groundstate $|\psi_0\rangle$ is invariant under the Z_2 "spin-flip" symmetry, $|\psi_0\rangle = \otimes_i^N \sigma_i^x |\psi_0\rangle$. In the ordered phase the Z_2 symmetry is broken which is a consequence of the degeneracy of the ground state of the ordered phase. As an example, both states $|\psi_1\rangle = |\uparrow\uparrow\uparrow\cdots\rangle$ and $|\psi_2\rangle = |\downarrow\downarrow\downarrow\cdots\rangle$ are groundstates of the system for $h = 0$. They are not invariant under a Z_2 transformation $|\psi_2\rangle = \otimes_i^N \sigma_i^x |\psi_1\rangle \neq |\psi_1\rangle$ [81]. Therefore, the system has to "choose" a groundstate when it undergoes a phase transition from the disordered symmetric state to the ordered symmetry broken state. This is what spontaneous symmetry breaking means.

An important question that we will investigate in this thesis is how ML can help us to identify phases of matter and how we can interpret ML predictions and to eventually include them in our research. To achieve this we will have to introduce a reformulation of a phase transition phenomena to a ML task. To do so we conclude this brief overview over the most important concepts of phase transitions with a very general description of what a phase and a phase transition is. A phase is a (equilibrium) state of matter that does not qualitatively change for small changes in external parameters and within this phase the thermodynamical potential is an analytic function of the external parameters. A phase transition is a qualitative change in the state of the system, where the thermodynamical potential becomes unstable with respect to the external parameters and behaves non-analytically. For QPT instead of the thermodynamical potential the ground state energy becomes a non-analytic function of the non-thermal external parameters.

Reformulation of a phase transition into a ML task

In this thesis we will look at a shift in paradigm in many-body physics that was first introduced by [10] where the authors reformulate the problem of finding a phase transition into a supervised ML task. The task is to distinguish the ferromagnetic from the paramagnetic phase of the classical 2D Ising model with nearest neighbour interactions. For this purpose the two phases were labeled and a fully connected NN was trained on spin configurations of both phases. What probably started as a pure curiosity if the ML algorithms are capable of finding these transitions evolved to a much more fundamental goal which is often referred to as a self-driving laboratory [82], which in the case of many-body physics would be an algorithm that can find new materials and phases of matter without human input. To automate the discovery of phase transitions it will be necessary to formulate the problem in an unsupervised manner or to use transfer learning to train a ML algorithm one a well known model and use it to find phase transitions on an other model. We use both concepts and study them extensively in the Chapters 3 and 4.

Apart from automatizing research, avoiding human input has another important reason. We as humans can bias the decision making process of the ML algorithms by introducing a priori knowledge about the underlying physics of the system and especially in the case of MBL phase transitions research has shown that the order parameters that were engineered by physicists do not even agree within error bars [83]. This leads to the question that the order parameters chosen by physicists might not be the most efficient approach to finding the exact critical point and that instead of telling the machine what kind of order it has to look for, we let the algorithm learn and decide itself what part of the input data is most relevant to find phase transitions. The interaction of a researcher with a ML algorithm and how we prepare the data is a crucial step and it will eventually also bias the outcome of the ML training and the predictions of the algorithm. Bias can occur from the choice of the input data. There is a famous example of a ML algorithm that was supposed to distinguish wolves from huskies. With interpretability methods the researchers found that most images from wolves were taken in a snowy environment and the ML algorithm only learned the snowy background of the photographs [84] and therefore did not learn to distinguish the animals. Clearly, this is a rather extreme example and in physics the bias can be more subtle. For example when the researcher already knows the physical system well and uses the known order parameters as an input. This will most likely prevent the discovery of new physics, because the ML algorithm will not be able to extract new or unknown qualitative changes in the data.

We can think of quantum states or measurements coming from a quantum system, matching a particular choice of parameters, as data instances with a label which indicates the corresponding phase if we intend to do a supervised ML setting. The discrimination of the phases can happen via supervised training, when the phases are known in advance, or via unsupervised training, when the phases are unknown. The latter is clearly harder, but it is also more interesting from a physics perspective, since it would allow us to map out an unknown phase diagram. In Chapter 3 we use a supervised approach, where we label quantum states coming from deep inside the phase, where one has some certainty of which phase a state is coming from and after the training the NN labels the test states close to the transition to find the critical parameter. In Chapter 4 we discuss a fully unsupervised method, that can map out the whole phase space after being trained on a set of quantum states that come from an arbitrary region of the phase space.

Different strategies can as well be adopted for the choice of the inputs of the NN. The first one would be to feed the order parameter or several order parameters to the machine and let it find the phase transition points. This approach is very intuitive for physicists but its main weakness is the requirement of ad-hoc engineering as one has to know which are the relevant order parameters. As described before, this could be a possible source for bias. The second one would be to feed directly the ground state of the Hamiltonian to the algorithm and let the machine itself discover the order parameters and the phase transition points. In Chapter 3 we follow the second strategy. Namely, we use the wavefunction coming from exact diagonalization as and input. In Chapter 4 we don't have access to the full wavefunction because we use the density matrix renormalization group (DMRG) method. Therefore, we use the entanglement spectrum as an input. In our opinion the wavefunction is the most unprocessed input data that is accessible from simulations. If exact diagonalization is not possible and the data has to be obtained by DMRG we use the entanglement spectrum. It is directly accessible from DMRG data and it is unbiased in the sense that it does not presume physical knowledge of the system but it is the mere spectrum of the reduced density matrix $\rho_A = \mathrm{Tr}_B \left|\psi\right\rangle \left\langle\psi\right|$ of the system $\left|\psi\right\rangle$. We will also study the use of the whole (central) tensor of a matrix product state from a DMRG simulation and we will also test certain ML methods with correlator data that are in theory accessible from experiments. The latter case is discussed in Chapter 4 and it becomes evident that the choice of measurements in an experiment already gives a strong bias on what phase transitions can be detected because for example topological phase transitions become invisible to the ML algorithm if the measurement is local.

Recent works have shown the feasibility of supervised and unsupervised phase detection. Standard unsupervised methods, such as principal component analysis or t-Distributed Stochastic Neighbor Embedding (t-SNE), have been used to characterize phase transitions in several systems such as Ising model, the XY model or the Hubbard model [19, 17, 42]. Other works used shallow NNs, i.e. fully connected NNs with a few layers, to characterize models such as the Ising model or the Bose–Hubbard model [11, 15, 17, 19]. The latter approaches all used fully connected learners, which do not scale well with the input size and depth of the network and have limited ability to extract features from the input [85]. Therefore the input of the NN had to be either small or hand-crafted, as in the case of using correlation functions.

This stands in contrast to deep learning, that revolutionized ML by providing automated means of extracting high-quality feature spaces from raw data [86]. Deep learning networks, however, struggle with the unsupervised scenario, and they are mainly applied in supervised problems. A body of work studied classical [10, 12, 13, 14] and quantum [15, 18, 19] phase transitions, and even topological phases [20, 21, 22, 23] with deep supervised architectures. Since automated feature extraction is desirable to investigate more complex systems, a few recent works ventured into using unsupervised deep learning techniques for studying phase transitions. Boltzmann machines are a computationally expensive, but highly expressive method [87], and computationally efficient feedforward CNNs can be tweaked in some cases to perform unsupervised learning [16] or transfer learning [17]. Furthermore, there is an unsupervised method called learning by confusion [11] which has the main drawback that it has to be trained many times.

We can conclude that ideally one leverages deep convolutional NNs on phase transition problems, with input data that is not biased and therefore not engineered. Instead of using shallow learners one lets deep architecture extract features automatically. The convolutional input layers are advantageous in reducing the dimensionality of the input state vectors and therefore hand-crafted dimensionality reduction can be avoided.

2.3 Variational Quantum Circuits and Quantum Neural Networks

In the chapter 6 we study the loss landscape of variational quantum circuits (VQCs) and Quantum neural networks (QNNs) with the Hessian. We introduce here the basics of VQCs and leave the details of how to obtain a gradient of a quantum circuit and the exact architectures that we use to the discussion in chapter 6.

Variational Quantum Circuits

The wave function contains all the information of a quantum state, but in general an exponential amount of classical information is needed to encode it. Because of this "curse of dimensionality" that makes the wave function grow exponentially with any additional spin or particle of a quantum many-body system there is a plethora of techniques how to approximate quantum states with classical representations. Just to name a few, there are quantum Monte Carlo (QMC) methods [88, 89, 90], matrix product states (MPS) [91, 92, 93], general tensor networks [94, 95] and in recent years NNs have been used as an ansatz for the wavefunction [96, 97]. These methods all have in common that they rely on unnormalized parametrized wave functions $|\psi(\boldsymbol{\theta})\rangle$ and optimize these parameters to minimize the energy

$$E(\boldsymbol{\theta}) = \frac{\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle}{\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta})\rangle} \tag{2.4}$$

to eventually find an approximation of the ground state of the system described by the Hamiltonian H.

In recent years NISQ devices [27] have become accessible and one of their most promising applications are VQCs. Since NISQ devices don't scale well and are still too noisy, it is not possible to run quantum algorithms on them that require error corrected qubits, like Shor's algorithm [98] for integer factorization or Grover's algorithm [99] to search a database. Instead, NISQ devices can be used to implement parametrized quantum circuits from which one can obtain measurements that would be in general hard to simulate on a classical computer. The idea of a VQC is, instead of parametrizing the wavefunction with classical objects such as MPS or NNs, one uses a quantum circuit with gates that are parametrized with the free parameters $\boldsymbol{\theta}$ and measures

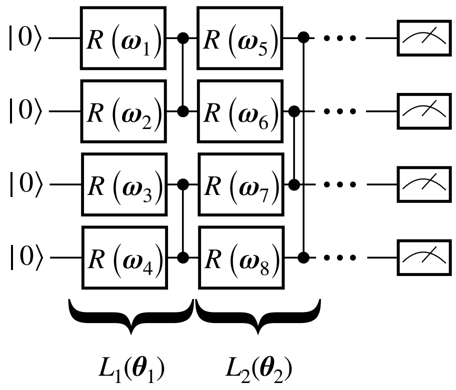


Figure 2.5: A way to approximate a general unitary is to use general SU(2) qubit rotation gates $R(\boldsymbol{\omega}) = R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$ followed by CZ gates and repeat it several times. A layer $L_i(\boldsymbol{\theta})$ is defined by $R(\boldsymbol{\omega})$ gates applied to each qubit followed by CZ gates that create entanglement.

observables that allow to calculate the energy. With the same optimization techniques as in classical ML tasks, one can optimize the parameters $\boldsymbol{\theta}$ to find low energy states. These methods are often referred to as quantum-classical hybrid optimizations, because the quantum circuit is only used to sample the measurements and the optimization itself is done classically.

In the NISQ devices that are currently available the parametrized gates are often limited to single qubit Pauli rotations $R_i(\phi) = \exp(-i\phi/2\sigma_i)$, where σ_i are the Pauli spin-1/2 matrices, with $i \in \{x, y, z\}$. To approximate a general n-qubit unitary one can use combinations of single qubit rotations followed by 2-qubit operations such as a controlled NOT (CNOT) or a controlled Z (CZ) gate like in Figure 2.5 and repeat this pattern several times. A general n-qubit unitary would require $\mathcal{O}(2^n)$ of one and two qubit gates [100] and is not feasible on current devices. A possible way to circumvent the growth of the circuit ansatz is to choose an architecture that is well suited for certain classes of problems, such as for example the Quantum Approximate Optimization Algorithm (QAOA) ansatz [101]. In chapter 6 we will look at simple combinations of rotational gates and 2-qubit gates.

Independent of the circuit ansatz we refer to the parametrized circuit as $V(\boldsymbol{\theta})$ and the variational state is given by $|\psi(\boldsymbol{\theta})\rangle = V(\boldsymbol{\theta})\,|\mathbf{0}\rangle$, where $|\mathbf{0}\rangle = |0\rangle^{\otimes n}$ is the standard n-qubit initial state of the device. With the aforementioned quantum-classical hybrid optimization one can optimize the expectation value $\langle\mathcal{O}\rangle = \langle\psi(\boldsymbol{\theta})|\mathcal{O}|\psi(\boldsymbol{\theta})\rangle$ of any measurable observable $\mathcal{O}$, the energy of the state $\langle E(\boldsymbol{\theta})\rangle = |\,\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle\,|$ or as well just try to maximize the fidelity $\langle\psi(\boldsymbol{\theta})|\psi_T\rangle$ with a possible target state $|\psi_T\rangle$.

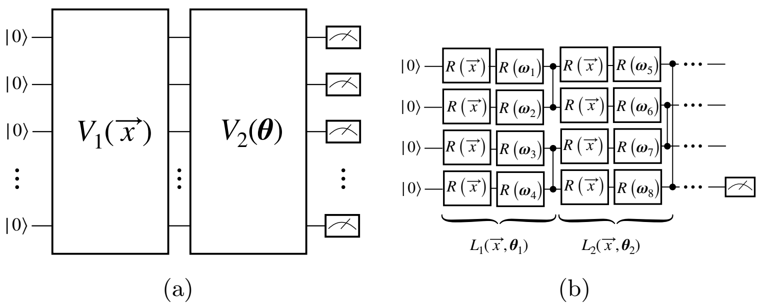


Figure 2.6: Possible realizations of QNNs: a) A unitary $V_1(\boldsymbol{x})$ encodes the classical data into a quantum state and the free parameters $\boldsymbol{\theta}$ of the unitary $V_2(\boldsymbol{\theta})$ are adjusted until we obtain a certain measurement. b) The reuploading scheme from [39] is constructed by layers of general single qubit rotations $R(\boldsymbol{\omega}) = R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$ with free parameters $\boldsymbol{\omega_i}$ and the input data $\boldsymbol{x}$ as their rotation angles. For a classification task with only two classes a single qubit measurement is sufficient and the measurement outcomes $\langle \sigma_Z \rangle = \pm 1$ are interpreted as the label prediction of the QNN.

Quantum Neural Networks

QNN are VQCs where parts of the parameters $\boldsymbol{\theta}$ are not free, but dependent on input data x_i coming from a data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_i^N$. The variational state takes the form $|\psi(\boldsymbol{\theta}, \boldsymbol{x}_i)\rangle = V(\boldsymbol{\theta}, \boldsymbol{x}_i)\,|\boldsymbol{0}\rangle$. Instead of optimizing the energy of such a state we reinterpret the measurement of an observable as the prediction of a label of the QNN. For example one can measure the expectation value

$$\langle \psi(\boldsymbol{\theta}, \boldsymbol{x}_i)|\, \sigma_Z \,|\psi(\boldsymbol{\theta}, \boldsymbol{x}_i)\rangle = \langle \sigma_Z \rangle_{\boldsymbol{\theta}, \boldsymbol{x}_i} \tag{2.5}$$

on one of the qubits and interpret the measurement outcomes $\langle \sigma_Z \rangle_{\boldsymbol{\theta}, \boldsymbol{x}_i} \in [-1, 1]$ as the prediction y' for the input data $\boldsymbol{x}_i$. Analogous to the classical NN a loss $l(y', y)$ can be defined and the ideal parameters $\tilde{\boldsymbol{\theta}}$ can be found via the minimization of the empirical risk

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} l(\langle \sigma_Z \rangle_{\boldsymbol{\theta}, \boldsymbol{x}}, y) \tag{2.6}$$

Figure 2.6(a) shows an example for a QNN architecture [102] where the unitary $V(\boldsymbol{\theta}, \boldsymbol{x}_i) = V_1(\boldsymbol{\theta})V(\boldsymbol{x}_i)$ is separated in a unitary that depends on the data $\boldsymbol{x}$ and another part that depends on free parameters $\boldsymbol{\theta}$. The first part of the circuit is used to prepare a quantum state from the classical input data $\boldsymbol{x}$ with a unitary $V_1(\boldsymbol{x})$ that returns a state $V_1(\boldsymbol{x})\,|\mathbf{0}\rangle = |\psi(\boldsymbol{x})\rangle$. This part is often referred to as state preparation. The parameters $\boldsymbol{\theta}$ of the unitary $V_2(\boldsymbol{\theta})$ are free and have to be optimized by the training scheme.

Figure 2.6(b) shows a different approach of how to encode the data into the quantum state $|\psi(\boldsymbol{\theta}, \boldsymbol{x}_i)\rangle$. In this scheme an alternating pattern between single qubit rotations $R(\boldsymbol{x})$ that depend on the data and $R(\boldsymbol{\omega})$ that depend on free parameters encode the data inputs onto the quantum state. The subfield of the study of encoding strategies drew recently a lot of attention [103] and shows promising result that might lead to a better understanding of QNNs. The specific architecture that we use in this thesis is shown in Figure 2.6(b) and it will be discussed in detail in Chapter 6.

Chapter 3

Domain Adversarial Phase Detection

A phase diagram shows qualitative changes in many-body systems as functions of the parameters of the physical system and the task of physicists is to identify the correct order parameter that captures this change. As discussed in 2.2, in the Landau theory of phase transitions, a discontinuity of the local order parameter or of one of its derivatives indicates a phase transition. In more exotic systems, the order parameters are global as it is the case for topological phases or topological insulators. The search of the *right* order parameters and the derivation of the phase diagram in terms of the parameters of the Hamiltonian prove to be very challenging tasks. Already for non-interacting Hamiltonians, where the addition of disorder or quasiperiodic disorder can lead to Anderson localization [104, 105] or to topological phase transitions [106, 107], distinguishing the phases can be demanding. The interplay of disorder and interactions can even give rise to many-body localization (MBL) [108] which is a phase transition not at thermal equilibrium and is, therefore, not captured by Landau's theory.

As described in Section 2.2 we can reformulate the distinction of phases of matter as a ML task, where the aim is to discriminate data instances with different labels or even in an unsupervised manner. In this Chapter we deploy a supervised method that only relies on labeled quantum states coming from deep inside the phase.

An example where ML assistance is much needed is the delineation and characterization of the aforementioned MBL phase, exhibited by systems with many interacting quantum particles experiencing a (strong enough) static disordered background potential. This research problem has attracted an immense amount of attention recently [109, 110, 111, 112, 113, 114, 115, 116] because MBL challenges long-held believes about the phase structure of isolated systems and even the applicability of standard equilibrium statistical mechanics, which no longer correctly captures the long-time behavior in that phase. Many details of how this breakdown happens remain elusive, despite the known characterization of MBL in terms of local conserved quantities [117, 118] and an extensive and ongoing debate [111, 114, 113, 116].

Traditionally, physicists have to identify the relevant order parameters for the classification of the different phases. We here in this Chapter follow a radically different approach: we address this problem with a state-of-the-art domain adaptation method called Domain Adversarial Neural Network (DANN) to derive the phase diagram of the whole parameter space starting from a fixed and known subspace. Through an adversarial approach, the DANN is capable of extracting invariant features of two different domains.

We benchmark the DANN method on two models. First we use the SSH model with disorder to benchmark the DANN's capability to do domain adaptation between state vectors coming from a Hamiltonian with and without disorder. Later we extend the SSH model to long range interaction because in this model new winding numbers appear and therefore, we can use the long range interaction to see if the invariant feature extraction also works for new phases. The second model that we investigate is the spin-$1/2$ Heisenberg chain in a random magnetic field that experiences a MBL phase transition. The capability of the DANN to extract invariant features from noisy data and, therefore, capture the relevant physics of the phase transition allows to "invent" a new "order parameter" for the MBL phase transition that yields meaningful results. Furthermore we show that the DANN can extract these features from vastly fewer disorder realizations than established methods.

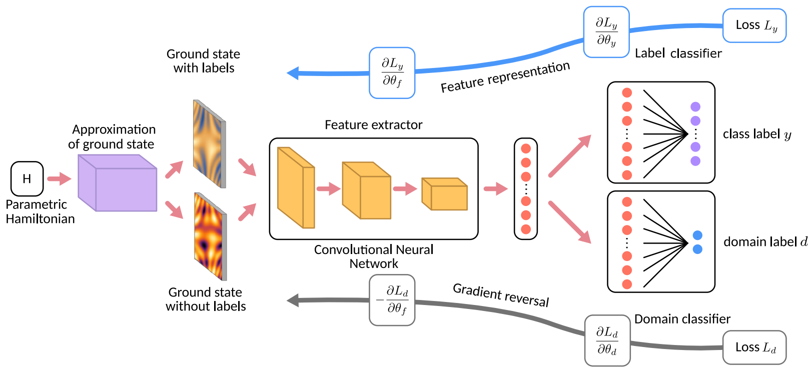
This approach avoids ad-hoc feature engineering and does not make assumptions about the input data, relying on deep learning to extract an expressive feature space. Furthermore our architecture allows to scale the size of the input because of the CNN. We show that in both cases, for the SSH model and the spin-$1/2$

Heisenberg model, the DANN shows advantageous behaviour over other techniques. The knowledge transfer for the SSH model from data without noise to data with noise does not work with NNs without domain adaptation, but it shows good results with the DANN architecture. For the MBL transition we address two major roadblocks preventing further progress: First, it yet remains unclear what the best approach is to delineate the MBL phase. Physicists have come up with a whole zoo of quantities whose behavior can serve as an indicator for the transition, but the various phase boundaries they imply, do not agree within error bars [112] and controlling finite-size effects is a challenge [119]. Second, all of these quantities need to be averaged over an enormous number of disorder realizations (often 10.000 [110, 112, 120]) to get meaningful results. Highly optimized codes allow in principle to study systems of up to 26 spins [115], but with the known quantities, going beyond 22 spins is prohibitively expensive because of disorder averaging [115]. We show that the DANN can capture the relevant features with up to 100x fewer disorder realizations to obtain objective and more accurate predictions of the transition point.

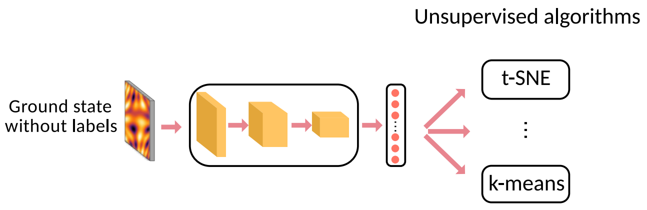
The rest of this Chapter is structured as follows. Section 3.1 reviews the idea of domain adversarial adaptation and discusses how this algorithm can be a powerful tool for the classification of phase transitions. In Section 3.2 we give an introduction into the SSH model, and Section 3.3 demonstrates the efficiency of the technique on the (SSH) model with disorder and long range hopping. Section 3.4 gives an introduction into the spin-1/2 Heisenberg model in a random magnetic field and the MBL phase transition and Section 3.5 shows how the DANN can be used to find a MBL phase transition with vastly fewer disorder realizations than other, non-ML based methods. Finally, Section 3.6 is dedicated to conclude the chapter.

3.1 Methods: Domain adversarial neural networks

The DANN is a deep learning method to do domain adaptation—also known as adversarial domain adaptation—where the feature extraction layers, consisting of a CNN, are trained to be invariant between a supervised source data distribution and a potentially unsupervised target data distribution [121]. We use a deep learning technique to let the ML algorithm decide what features are relevant to be in the domain invariant space and to introduce as little assumptions about the physics of the system as possible.


Ground state with labels
$\frac{\partial L_y}{\partial \theta_f}$
Feature representation
$\frac{\partial L_y}{\partial \theta_y}$
Label classifier
Loss L_y
Approximation of ground state
H
Parametric Hamiltonian
Feature extractor
Convolutional Neural Network
class label y
domain label d
Ground state without labels
$-\frac{\partial L_d}{\partial \theta_f}$
Gradient reversal
$\frac{\partial L_d}{\partial \theta_d}$
Domain classifier
Loss L_d

(a)



(b)

Figure 3.1: (colour) Schematic representation of our architecture. (a) Given a parametric Hamiltonian, we find the ground states sampled from two different distributions, e.g. noise levels of the Hamiltonian. For one of them—the source—we know the labels. For the other one—the target—we do not. A CNN is used as a feature extractor. The final layer of the representation is fed into a domain and a label classifier to find the correct phase labelling and to identify which domain the data comes from, respectively. The gradient reverse layer adds a negative constant to the back propagation of the domain classifier, which makes the feature distributions of the two domains similar. (b) We send the unlabelled examples across the trained feature extractor, and feed the high-dimensional representation to unsupervised learning methods to identify the phase transition.

As described in Section 2.1, for domain adaptation we have two types of input data distributions, which we refer to as different domains. In our case, these distributions come either from two different regions of the phase diagram of a model or from two different regimes of noise in the model. Concretely, in the case of the MBL phase transition one data distribution comes from deep inside the phases from a wide range of energies where one can be sure how to label the data. The other distribution comes from a region close to the phase transition where labelling is not possible. For the SSH model the well-known, labelled data distribution consists of eigenstates coming from a Hamiltonian without noise and the unknown data comes from a Hamiltonian with noise. The task of the NN is to learn from the labeled instances and adapt this knowledge to the new unknown instances coming from another distribution. Since the data is coming from two different domains we have to extract features from the data that are invariant under the domain. In other words, we need to extract only the features from the input data that are relevant for phase discrimination and not for domain discrimination. To achieve this, the DANN setup consists of three parts: the feature extractor, the label classifier, and the domain classifier as shown in Figure 3.1(a). To make the features invariant under the domain information, the DANN is trained such that the domain classifier cannot distinguish between them anymore. The label classifier on the other side is only trained on the source data. After the training, we predict the labels of the data from the target domain by feeding the inputs to the feature extractor and the label classifier, without the domain classifier. Alternatively we can as well not use the label classifier and apply unsupervised algorithms such as t-SNE [122, 123], k-means clustering [124] or density-based spatial clustering of applications with noise [125] directly on the feature representation.

In our case, the labelled input data set of a well known region of the phase diagram of the model is called the source data $\mathcal{D}_S = \{(\boldsymbol{x}_s, y_s)\}$ and comes from the source domain $\mathcal{S}$. The data set coming from an unknown region, without labels, is called the target data $\mathcal{D}_T = \{(\boldsymbol{x}_t)\}$ and comes from the target domain $\mathcal{T}$. Our goal is to predict the labels y_t for given inputs $\boldsymbol{x}_t$ of our target data. To distinguish whether the input $\boldsymbol{x}_i$ is coming from the source or target distribution, we introduce the domain label d_i, which is $d_i = 0$ if $\boldsymbol{x}_i$ is from our source distribution or $d_i = 1$ if $\boldsymbol{x}_i$ is from the target distribution. During the training of the DANN, we feed the input $\boldsymbol{x} \in \mathcal{S} \cup \mathcal{T}$ into the feature extractor where it is mapped to a high-dimensional feature vector $\mathbf{f} = G_f(\mathbf{x}, \boldsymbol{\theta}_f)$. This latent representation of the state vectors is forwarded to the label classifier $G_y(\boldsymbol{f}, \boldsymbol{\theta}_y)$ and the domain classifier $\boldsymbol{a} = G_d(\boldsymbol{f}, \boldsymbol{\theta}_d)$. The $\boldsymbol{\theta}_i$ represent the parameters that have to be learned through the training.

The feature extractor consists of CNNs, composed of many different filters. After a convolutional layer, we apply a max-pooling layer to further reduce the dimensionality of the input. Since there is only labelled data for the source part of the input $\boldsymbol{x}$, the loss of the label classifier can only be calculated by the source part of the feature vector $\mathbf{f}$. The loss of the domain classifiers can be calculated on all the inputs $\boldsymbol{x} \in \mathcal{S} \cup \mathcal{T}$.

To train the network we define the domain and classifier losses L_d, L_y. As described in Ref. [121], the domain classifier loss is a regularization of the label classifier. Therefore the training of the DANN optimizes

$$E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d) = L_y(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y) - L_d(\boldsymbol{\theta}_f, \boldsymbol{\theta}_d) \tag{3.1}$$

by finding the saddle point

$$(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y) = \underset{\boldsymbol{\theta}_f, \boldsymbol{\theta}_y}{\operatorname{argmin}} \; E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d) \tag{3.2}$$

$$(\boldsymbol{\theta}_d) = \underset{\boldsymbol{\theta}_d}{\operatorname{argmax}} \; E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d). \tag{3.3}$$

The update rule for the feature extractor therefore has the form

$$\boldsymbol{\theta}_f \leftarrow \boldsymbol{\theta}_f - \mu \left(\frac{\partial L_y}{\partial \boldsymbol{\theta}_f} - \frac{\partial L_d}{\partial \boldsymbol{\theta}_f} \right), \tag{3.4}$$

which can be implemented via stochastic gradient descent and the gradient reversal layer [121]. The domain classifier should not be able to distinguish the two domains because their feature representation is invariant. This is achieved by training the parameters of the domain classifier $\boldsymbol{\theta}_d$ such that the domain loss L_d is minimal. At the same time, the parameters $\boldsymbol{\theta}_f$ of the feature extractor are identified by minimizing the function $E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d)$. Since the domain loss also depends on the feature extraction parameters $\boldsymbol{\theta}_f$, this optimization problem has an adversarial character and leads to a competition between the optimization of the domain classifier and the label prediction loss or $E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_y, \boldsymbol{\theta}_d)$. This results in a domain classifier that is well trained, but is unable to distinguish the domains, as the feature representation of the two domains is invariant. For the label classifier's output, the training is similar except that both parameters $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_y$ minimize the classifier loss.

To predict the labels of the target distribution, we can either apply the label classifier or directly use unsupervised methods as t-SNE or k-means on the feature representation.

Input Data for the SSH model

The ground state of the non-interacting SSH model can be found via the diagonalization of the single particle Hamiltonian $H = \sum c_i^\dagger H_{i,j} c_j$. Here, we use a system size of 32 unit cells. The input data for the DANN are the fermionic occupied states, which are the states with negative energy eigenvalues and the zero energy state. We arrange these states in a matrix, where the eigenstates are the columns. Therefore the input data is two dimensional.

Input Data for the Heisenberg model

The spin-1/2 Heisenberg model has to be diagonalized in the full σ_Z basis and therefore ground states can only be obtained up to a system size of $N = 18$ spins. The input of the NN is the ground state vector of 2^N dimensions.

Details of the neural network architecture

For the SSH model the feature extractor of our DANN consists of two-dimensional convolutional layers each with 32 filters. For two dimensional inputs (SSH), the receptive field size is 3×3 and the pooling size is 2×2. The input size in 2D is for every model 64×64.

For the Heisenberg model the feature extractor of our DANN consists of four one-dimensional convolutional layers with four filters each and a filter length of 3. The input to the first layer is the ground state. Each layer has four of these filters, extracting different features of the input to the layer.

The fully connected layers of the NN are built in the same way for both physical models. The label classifier and the domain classifier have the same architecture and they contain 128 hidden ReLU neurons and 2 softmax output neurons. The difference between them is the gradient reversal layer between the feature extractor and the domain classifier. The activation function of these layers are rectified linear units (ReLUs). This is a piecewise linear function that outputs zero for negative values and a linear response for positive values. While NNs traditionally used nonlinear activation functions, the ReLUs have better numerical properties when training the network with many layers. Each of those layers is followed by a max-pooling layer that pools from three neighboring neurons. This is a critical step for coarse-graining the representation: we pick the maximum of the value of the activation over three neighboring points and discard the other two. In effect, we reduce the dimension of the vector by two-thirds in each of these pooling layers. Pooling does not only ensure a lower-dimensional representation, but it also enables that the subsequent convolutional layer identifies longer range correlations in the original data. We apply batch normalization after every layer, which introduces a slight stochastic variation in the scale of the characteristics of the input states, and thus reduces the chance of overfitting. Furthermore, for both models, we use dropout [126] for the fully connected layers in the phase discriminator and adversarial networks, which is standard practice in achieving better performance.

To ensure the reproducibility of our results, we made the source code available under an open source license [127] where one can find details about the exact NN architecture, pooling and learning rates.

3.2 Methods: The SSH model

The SSH model with disorder

The SSH model is a relatively simple and well studied model that shows a topological phase transition. Because of this we chose to use this model to benchmark our NN to find out if it can predict global state properties such as the winding number. Furthermore, we can add noise to the inter site hopping amplitudes which allows us to study the NN prediction with noisy input data. Here we introduce some basics about the model. The results of the NN phase classification is discussed in Section 3.3.

The SSH model is a one-dimensional-chiral model that exhibits topological properties: this system is characterized by a global topological invariant, the winding number. The term chiral model refers to the chiral symmetry of the Hamiltonian $\Gamma H \Gamma = -H$, with a local unitary operator Γ. This chiral symmetry leads

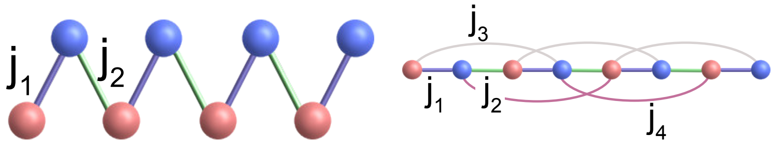


Figure 3.2: Caption

to a symmetric energy spectrum and therefore each eigenstate of the Hamiltonian $|\psi_+\rangle$ has a chiral partner $|\psi_-\rangle$ with the energies $\pm\epsilon(k)$ for a fermi energy $E_F = 0$ [128]. The winding number $\nu = \oint \frac{dk}{\pi} i \langle\psi_+|\psi'_-\rangle$ (with $|\psi'_-\rangle = \partial_k |\psi_-\rangle$) is either 0 or 1, being $\nu = 1$ in the topological non-trivial phase [129].

The Hamiltonian of non-interacting spinless fermions reads

$$H_{SSH} = \sum_n j_{1,n} c_n^\dagger \sigma_1 c_n + j_{2,n} \left[c_n^\dagger \sigma_+ c_{n+1} + \text{h.c.} \right], \tag{3.5}$$

with the pauli matrices σ_i and $\sigma_+ = \sigma_1 + i\sigma_2$. The SSH model is a tight-binding model with a two site unit cell. $j_{1,n}$ is the hopping amplitude within the unit cell and $j_{2,n}$ is the hopping amplitude for between the unit cells (See also Figure 3.2(a)). We study this model at half filling, which means we have one electron per unit cell. This model describes a 1D chain of two different types of alternating nuclei that are coupled and the electrons can move along the chain.

The disorder can be added in the hopping parameters $j_{1,n} = j_1 + W_1\omega_n$ and $j_{2,n} = j_2 + W_2\omega'_n$, where ω_n and ω'_n are randomly distributed numbers in the interval $[-0.5, 0.5]$. And W_1 and W_2 indicate the disorder strength of the two hopping amplitudes j_1 and j_2.

A phase transition from the topologically trivial ($\nu = 0$) to a topological phase ($\nu = 1$) occurs at $j_2 = j_1$ with the topological phase in $j_2 > j_1$.

Periodic and open boundary conditions

We study the SSH model with open and periodic boundary conditions (OBC and PBC). For PBC we set $j_{2,N} = j_{2,0}$ for N unit cells, which connects the $N - th$ unit cell with the first one. The winding number does not depend on the boundary conditions, but for OBC one obtains edge states for $\nu = 1$ at zero energy.

SSH model with long range hopping

We also consider the SSH model with nearest-neighbour hopping j_1 and j_2, and third-nearest neighbour hopping j_3 and j_4, as shown in Figure 3.2(b), which has the Hamiltonian

$$H = H_{SSH} + \sum_n j_{3,n} c_n^\dagger \sigma_1 c_{n+1} + j_{4,n} \left[c_n^\dagger \sigma_+ c_{n+2} + \text{h.c.} \right]. \quad (3.6)$$

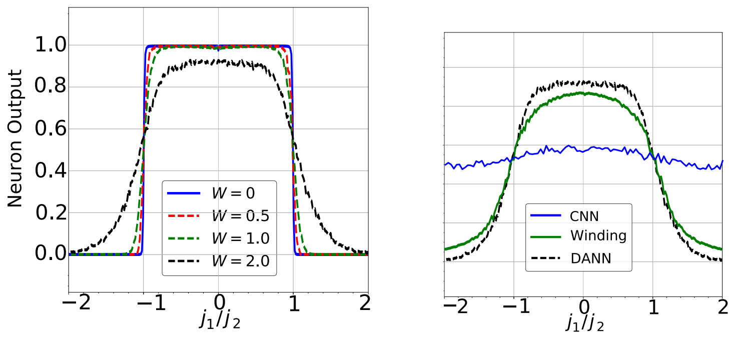
In this case, the phase diagram becomes richer with higher winding numbers [130]. By considering third-nearest neighbour hopping j_3 and j_4, additionally to the winding numbers $\nu = 0,\ 1$, we can also obtain winding number $\nu = \pm 1, \pm 2$. We study this model to find out if domain adaptation is able to transfer knowledge of the short range SSH model to new unknown phases with higher winding numbers.

3.3 Results: Domain adaptation with SSH

To train the DANN we generate source states with no disorder $W = 0$ and label them analytically with the winding number. The states in the topologically trivial phase have label 0 and the states in the topological phase have label 1. We then generate a target dataset with $W_1 = 2W_2 = W$, where the correct labelling is unknown. For increasing disorder the winding number and the DANN output is averaged over several disorder realizations.

Open Boundary conditions

We first apply the algorithm for the system with open boundary conditions. Figure 3.3(a) shows the classifier output for different disorder strengths averaged over 1000 disorder realizations. We correctly identify a shift of the topological phase transition with increasing disorder, which is in accordance with Ref. [107]. Furthermore, we compared the phase transition points with the one obtained from the winding number defined in Ref. [107], shown in Figure 3.3(b). Remarkably, the DANN predicts precisely the transition point.



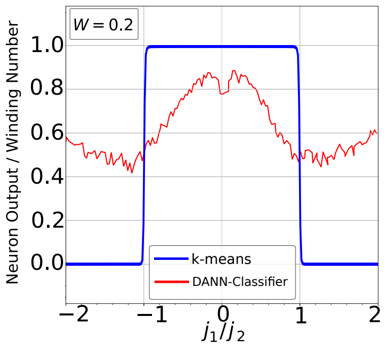
(a) (b)

Figure 3.3: SSH with open boundary conditions. (a) Neuron output of the SSH model for different disorder strength W and for a system of 64 sites with open boundary conditions. The results are averaged over 1000 disorder realizations. (b) Comparison of the phase predictions of a convolutional NN without domain adaptation (CNN), with the domain adversarial approach (DANN) and with the winding number for different values of j_1 and fixed $W = 2$. While the transfer learning fails, the phase transitions predicted by the DANN are in good agreement with the winding number.

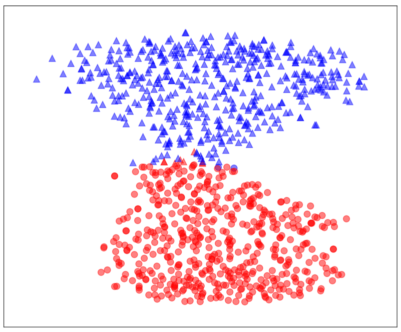
We also compare the DANN prediction with a CNN with the same label classifier and feature extractor but without the domain adaptation. To compare the efficiency of domain adversarial adaptation to the one without domain adaptation, we train a NN composed of a feature extractor and a classifier on the states without disorder. The architecture of the feature extractor and the classifier is chosen to be the same as the one of the domain adaptation. Figure 3.3(b) compares the predictions of the phase diagram of the SSH model with disorder $W = 2.0$ without domain adaptation (blue), with domain adaptation (dashed dark) and with the winding number (green). In this case, the CNN is not able to predict the label of input states with disorder $W = 2.0$, where the DANN is in good agreement with the winding number. This shows that the DANN is able to extract the relevant features from the noisy input states and a conventional CNN is not.

Periodic Boundary conditions

We then focus on the case of periodic boundary conditions. Here, in the absence of edge states the label classifier struggles to accurately predict the phase transition, as presented in Fig. 3.4(a). This is related to the fact that, within periodic boundary conditions, the classifier has to find a global property of the bulk of the system. Nevertheless, we can still perform unsupervised learning directly on the invariant feature representation. We first apply the t-SNE algorithm [122] which allows us to reduce the dimension of the feature representation to two. Figure 3.4 shows the t-SNE plot for one realization of disorder $W = 0.2$. The trivial (circles) and topological states (triangles) form two clearly separated clusters that can be labelled with k-means clustering. This method allows us to find the phase transition with good accuracy.



(a)



(b)

Figure 3.4: (colour) (a) k-means classification applied on the feature space of the DANN trained on SSH model with periodic boundary conditions compared to the DANN label classifier output. For periodic boundaries. The classifier of the trained DANN can not distinguish the phases of states with disorder. (b) Clustering of the two phases with t-SNE. The shapes indicate the correct labelling, the colours show the labelling found by k-means. If we apply k-means clustering on the low-dimensional embedding provided by t-SNE on the feature space, the labelling works well and the phase boundary can be found with an accuracy of $j_1/j_2 = 1 \pm 0.01$. The plots shows the SSH model with periodic boundary conditions with disorder strength $W = 0.2$

SSH model with long range hopping

The SSH model with long range hopping shows new winding numbers and our purpose is to see whether our scheme allows one to predict these new, unseen phases. As before, we generate source states for the SSH model for $j_2 = 1$, $j_3 = j_4 = 0$ and label them analytically with windings 0 and 1. We then produce target states for the SSH with long range hopping for $j_2 = j_1 = 1$ and $j_3 = 0$. Although the classifier has been trained to distinguish data points with windings 0 and 1, it accurately detects phase transitions between trivial and topological phases, as shown in Figure 3.5(a) (solid line), even if the topological phase is of higher winding number $\nu = 2$ that has not been part of the training. Furthermore, when analysing the feature space directly, additionally to the clustering trivial / topological phases we find a subclustering in the topological phase that separates $\nu = 1$ from $\nu = 2$. K-means can predict the labels of the trivial phase ($\nu = 0$) with high accuracy. The transition between winding numbers $\nu = 1$ and $\nu = 2$, on the other hand, is not accurate close to the phase transition, as shown in Figure 3.5(a) (dashed line). Nevertheless, far from the phase transition, the k-means algorithm labels the phases correctly.

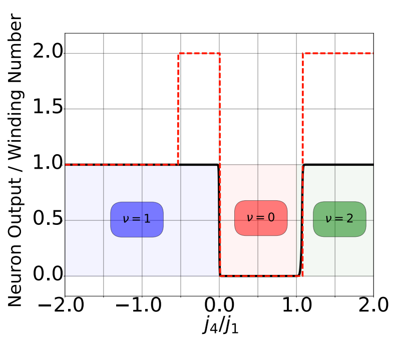
3.4 Methods: Many-Body Localization

After successfully benchmarking the DANN architecture on the SSH model, we study the problem of delineating the MBL phase boundary in the prototypical spin-1/2 Heisenberg chain in a random magnetic field, described by the Hamiltonian

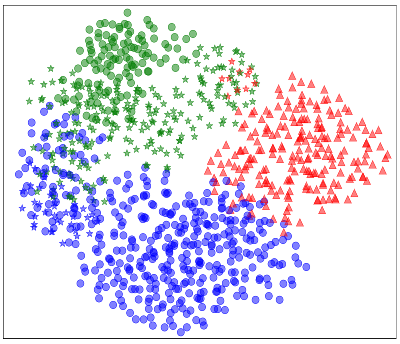
$$H = \frac{1}{2} \sum_{i=1}^{N} \sum_{\alpha \in \{x,y,z\}} \sigma_i^{\alpha} \sigma_{i+1}^{\alpha} - \sum_{i=1}^{N} h_i \sigma_i^z, \tag{3.7}$$

with $\sigma_i^{x,y,z}$ the Pauli matrices on site i and the h_i are drawn from the uniform distribution over $[-h, h]$.

Even though closed quantum systems follow unitary time evolution, at low enough disorder they have been shown to be able to thermalize [131], which is often referred to as that they satisfy the eigenstate thermalization hypothesis (ETH) [132]. In a nutshell, for a system that satisfies the ETH the reduced density matrix of a subsystem A $\rho_A = \mathrm{Tr}_B(|n\rangle \langle n|)$ where B is the complement



(a)



(b)

Figure 3.5: (colour) SSH Long Range label prediction of the classifier (solid line). If we fix $j_1 = j_2 = 1$ and $j_3 = 0$ we can find phase transitions $\nu = 1 \rightarrow 0$ at $j_4 = 0$ and $\nu = 0 \rightarrow 2$ at $j_4 = 1$ [130]. The dashed line shows the labelling found by k-means directly on the feature space. We can see that there is a mislabelling at the boundary between $\nu = 1$ and $\nu = 0$. In colour are the effective winding numbers. j_1 and j_2 are the nearest neighbour hopping terms, j_3 and j_4 are the third nearest neighbour hopping terms. In the right panel is the SSH Long Range feature space classification via k-means and graphical embedding by t-SNE. The shapes indicate the correct labelling, the colours show the labelling found by k-means.

of A will take a thermal form $\rho_A(|n\rangle) \approx \exp(-H/T_n)$. The state $|n\rangle$ is an eigenstate of H with energy E_n. The temperature T_n is given by the canonical ensemble that reproduces eigenstate energy $\langle H\rangle_{T_n} = \langle n|H|n\rangle$. For strong disorder the eigenstate behave differently. For the extreme case where the local fields $h \rightarrow \infty$ the eigenstates take the form of a product state $|\uparrow\uparrow\downarrow\uparrow\ldots\rangle$ and the ETH breaks down. The eigenstates of the spin-1/2 Heisenberg model in a random magnetic field are known to undergo an MBL transition at an energy dependent critical disorder strength h_c, whose precise position is however difficult to determine with established methods. The most widely used method to detect the MBL transition is the average adjacent gap ratio r [112, 120], which goes from $r_{\mathrm{WD}} \approx 0.53$, resulting from the Wigner-Dyson distributed eigenvalues in the ergodic phase, to $r_{\mathrm{Poisson}} \approx 0.38$, reflecting the Poisson statistics in the MBL phase.

To determine the average adjacent gap ratio we first denote the normalized energy by $\epsilon = (E - E_{\max})/(E_0 - E_{\max}) \in [0, 1]$, which interpolates between the lowest and highest of the energies of H for a given realization of the disordered fields h_i and we restrict the eigenstates to the global magnetization zero subspace $\langle S \rangle = 0$. For a given realization of h_i one takes the ratio of the level spacing of consecutive energy levels $r(n) = \min(\delta(n), \delta(n+1))/\max(\delta(n), \delta(n+1))$ with $\delta(n) = E_n - E_{n-1}$ at a given eigenenergy E_n and averages them over 50 eigenpairs with energies closest to the target energy ϵ. And finally one averages these values over several different disorder realizations h_i drawn from the interval $h_i \in [-h, h]$. This returns a single point in Figure 3.7(b) for a given h and ϵ.

There are other quantities to determine the phase transition, such as the dynamical spin fraction [110] or the entanglement entropy per site [112], but all these methods lead to different results that sometimes do not even agree within their error bars.

3.5 Results: Domain adaptation with MBL

For the training of the DANN we generate eigenstates from small windows around several values of the renormalized energy ϵ and for multiple disorder realizations at different disorder strengths h for system sizes up to $N = 18$ spins with the shift invert code from [115]. The set of states from deep inside the phases $\mathcal{D}$ was drawn from $h \in [0.1, 0.5]$ for the delocalized phase and from $h \in [7.0, 8.0]$ with a step size 0.1 for the MBL phase for energy densities in the range $\epsilon \in [0.05, 0.95]$. To have equally big sets in the delocalized and the MBL phase the values of h are separated by steps of 0.05 in the delocalized phase, 0.1 respectively in the MBL phase. The epsilon values are separated by steps of 0.05. For the set close to the phase boundary we choose states with disorder strength $h \in [0.5, 7.0]$ separated by steps of 0.2 and normalized energy $\epsilon \in [0.05, 0.95]$ in steps of 0.05. For each set of parameters and disorder realization we find the 50 states closest to the chosen renormalized energy ϵ. We take several realizations for each point in the parameter space which is chosen such that both sets are of the same dimension, namely 50k. We have checked that the results do not depend on the details of how these sets are chosen, to be sure that the success of the prediction is not dependent on the dataset. The states were produced with the open-source software from [115]. For the training of the network we use states from two sets deep inside the phases as the source dataset, which we can label (indicated in blue in Figure 3.6). For the target dataset we generate states from a wide range of ϵ and h values that including the phase boundary (indicated in red in Figure 3.6). We want to emphasize that we use the coefficients of the wavefunction as input data without further preprocessing.

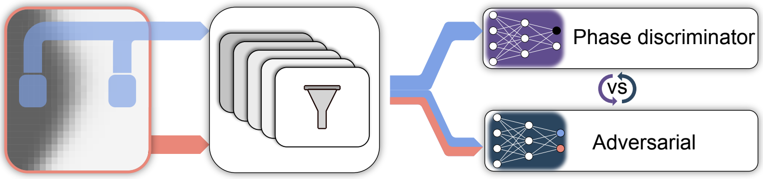
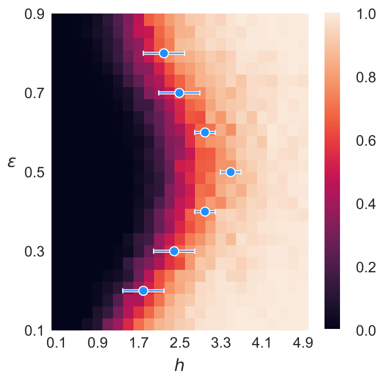


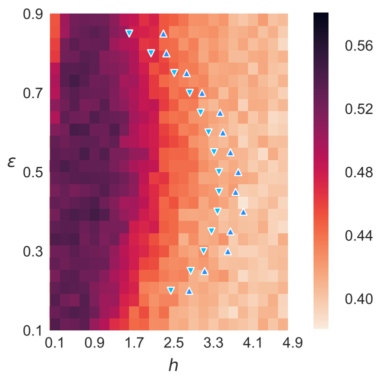
Figure 3.6: DANN architecture applied to MBL: The source set comes from deep inside the two phases, indicated in blue. The target set comes from the whole phase diagram, indicated in red. Both sets are fed through several layers of convolutional filters. The phase discriminator is solely trained on the source data (with class labels). And the Adversarial, or also called domain classifier, is trained on the source and target set with labels that indicate that an input vector comes from the source or the target set.

We compare the estimate of the energy resolved phase diagram obtained from the adversarial NN with results based on the average adjacent gap ratio r and the dynamical spin fraction f. The superior statistical properties of our approach are apparent. Already from only 50 disorder realizations we obtain a clear characterization of the phases, while the average adjacent gap ratio is still very noisy (background in Figure 3.7).

The phase boundary shown in Figure 3.7(a) can be determined via a data collapse from plots such as that shown in Figure 3.8 for $\epsilon = 0.5$ and in Figure 3.9 for all ϵ. To extract the critical magnetic field strength h_c, we use a scaling function of the same form $L^{1/\nu}(h - h_c)$ as that for the average adjacent gap ratio r [112]. The idea of a data collapse is to rescale the NN outputs with the rescaling function such that the curves for different system sizes L overlap exactly. From this procedure we can extract the values for ν and h_c. The reduction in noise allows for a more precise determination of the phase boundary for the same number of disorder realizations. From just data for systems up to size $N = 18$ (100 disorder realizations) we are able to determine the phase boundary extrapolated to the thermodynamic limit to an accuracy roughly matching the discrepancy between the conventional quantities r and f determined in the numerically much



(a)



(b)

Figure 3.7: The output of the NN directly provides a meaningful estimate of the phase diagram for a finite system $N = 12$ (background in (a)) from just 50 disorder realizations, while traditional quantities, like the gap statistics (background in (b)) are still far too noisy. The dots in (a) are the extrapolated phase boundary in the thermodynamic limit obtained from 100 disorder realizations via the data collapse for systems up to $N = 18$, shown exemplary for $\epsilon = 0.5$ in Figure 3.8 where we compare a data collapse of the DANN output with the data collapse of the average adjacent gap ratio for the same amount of disorder averaging. In Figure 3.9 we show the full data collapse of the DANN output for all ϵ. The symbols in (b) are the phase boundaries found in [112] based on the average adjacent gap ratio r (triangles) and the dynamical spin fraction f (triangles pointing downwards) for systems of size up to $N = 22$ and vastly more disorder realizations (the data collapse plots for all values of ϵ are shown in Figure 3.9).

more expensive study of [112]. Intuitively, it makes sense that the average adjacent gap ratio does not have the nice averaging properties of the quantity computed by our NN, as it completely disregards the properties of the eigenstates and only computes one feature of the spectrum. ML, in contrast, figures out a way to objectively determine the phase by directly recognizing non-trivial properties of the eigenstates.

Another interesting feature in which our method differs from average adjacent gap ratio (as well as most other quantities that have been used in exact diagonalization studies so far) is the value of the scaling exponent ν. Our method consistently yields $\nu \approx 1.6$, independent of the energy range and the precise choice of the training data (under the condition that it is sufficient to ensure convergence of the training), while the average adjacent gap ratio yields $\nu \approx 0.9$ [112]. Both exponents violate the (heuristic) Harris criterion, which for one spacial dimension predicts $\nu > 2$ [133], but the larger value of our "order parameter" is closer to the predicted value and there is hope that by moving to even larger system sizes, the best data collapse will be obtained with $\nu \approx 2$. This is another indication that our automatically detected "order parameter" suffer less from finite-size effects than more traditional quantities. The size of the region in which the network is unsure which label to assigned shrinks during training and eventually converges. It is a natural measure for the broadening of the phase transition due to finite-size effects.

Our method has a number of additional desirable properties. The intermediate values of the average adjacent gap ratio do not have a physical meaning, whereas the output of the NN has an immediate interpretation as to how certain the phase prediction is. The predicted values of h_c and the sizes of the plateaus are stable against changing the regions from which the first kind of training data is generated. The average adjacent gap ratio, actually attains the Poisson value at the integrable point at $h = 0$ and it moreover fails to capture the transition if one does not restrict to a fixed magnetization sector. Our method does not suffer from either of these two drawbacks.

Importantly, the computational time for training and evaluating the output of the adversarial NN is almost negligible compared to the time it takes to generate states for mapping out the phase diagram. As much fewer disorder realizations are necessary per point, this yields a huge net gain in computational time. Our

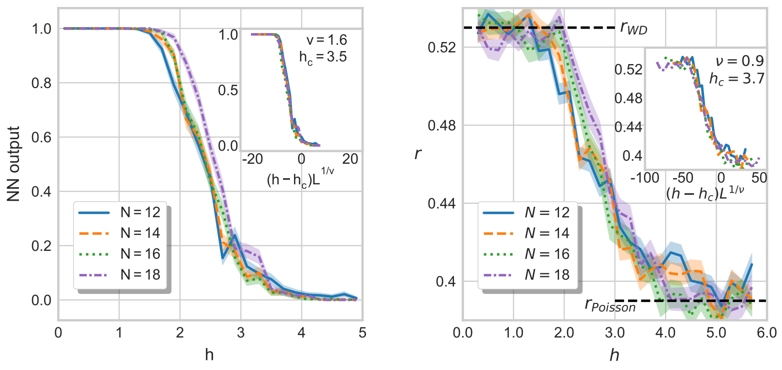
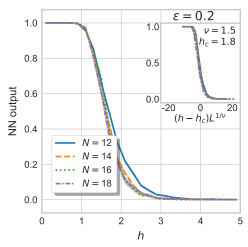


Figure 3.8: Exemplary output (a) of the NN at normalized energy $\epsilon = 0.5$ averaged over 50 disorder realizations and the data collapse (inset) to determine the position of the phase transition h_c in the thermodynamic limit. The average adjacent gap ratio r is still far too noisy (b) to obtain a good collapse (inset) for the same amount of averaging. The error bands show the ensemble standard deviation $s = (\sum_i^N (x_i - \hat{x})^2/(N-1))^{1/2}$ of the disorder average.

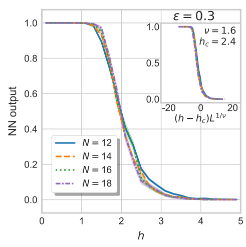
approach thus allows to meaningfully include states from larger system sizes, which can now be generated with state-of-the-art shift invert algorithms [115], into studies of MBL.

3.6 Conclusions

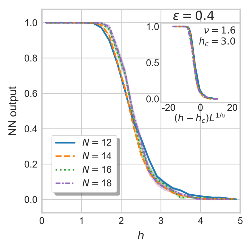
We have demonstrated that ML can be used to automate the task of identifying relevant features that most efficiently capture the physics of phase transitions in quantum systems — a formidable task so far reserved for human researchers. As humans, we often gain an intuition on a physical system using a special case that is analytically or numerically easy to treat. Then we generalize the insights to the more complex cases. Domain adaptation captures this idea: a deep learning system extracts intuition on a well-understood system and applies it to a more perplexing one. This is a subtle, targeted application of ML, with the explicit purpose of avoiding brute force numerical methods. We demonstrated the applicability of the method on the SSH model with a topological phase transition and on the spin-1/2 Heisenberg model in a random magnetic field with a MBL phase transition. For the SSH



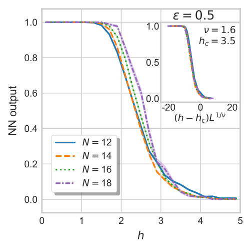
(a)



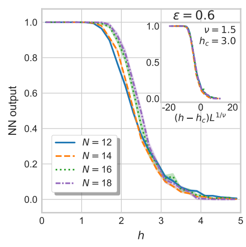
(b)



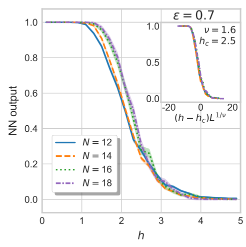
(c)



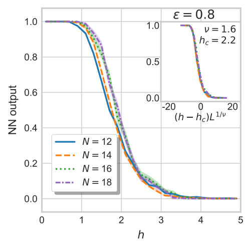
(d)



(e)



(f)



(g)

Figure 3.9: Output of the NN at energies $\epsilon = 0.2$ to $\epsilon = 0.8$. $N = 12$ and 14 are averaged over 500 disorder realizations and $N = 16$ and 18 over 100 realizations. The data collapse (inset) determines the position of the phase transition h_c in the thermodynamic limit. The error bands show the ensemble standard deviation $s = (\sum_i^N (x_i - \hat{x})^2/(N-1))^{1/2}$ of the disorder average. In Appendix A one can find the critical values and their errors in a tabular form.

model the phase diagram found by the algorithm is in very good agreement with the one obtained with standard methods and the algorithm can even predict new phases as shown in the long-range SSH model.

For the MBL phase transition the competitive process of adversarial domain adaptation, is able to "invent" a new "order parameter" that yields meaningful results from vastly fewer disorder realizations than established methods. It seems fair to say that the resulting quantity actually captures the essential physics, as the network, once trained, can correctly identify the phase transition not only at different energy densities, but also in similar but distinct models. This is remarkable, since the MBL transition has mostly defied analytical approaches and even the question of what is the best way to delineate the phase could not be resolved in a satisfactory way. As the automatic feature identification does not rely on a human understanding of the underlying physical processes, our approach has the potential to lead to new insights into poorly understood many-body phenomena such as MBL or topological phases through an analysis of the feature extraction layer.

Chapter 4

Anomaly Detection

In this Chapter, we demonstrate how to map out a phase diagram of a quantum many-body system to identify regions of interest for possible new phases using automated and unsupervised ML based on anomaly detection [46, 47, 48]. This approach is particularly useful when one is confronted with sufficient data from known classes of states and little or no data from unknown classes.

Compared to previous unsupervised attempts in [42, 12, 11, 19, 17, 23], this method needs only one or few training iterations and has better generalization properties from employing deep NNs [134, 135]. This allows for efficient fully automatized phase discovery in the spirit of self-driving laboratories [82], where artificial intelligence augments experimentation platforms to enable fully autonomous experimentation. Intuitively, the method explores the phase diagram until an abrupt change, an anomaly, is detected, singling out the presence of a phase transition. The intuition is similar to the approach introduced in [136], where the authors proposed to detect quantum phase transitions by looking at the fidelity between ground states $\mathcal{F}(p, \tilde{p})) = |\langle\psi_p|\psi_{\tilde{p}}\rangle|$, where p and $\tilde{p} = p + \delta p$ are neighbouring parameters in the phase diagram. The idea is to detect a transition because the phase boundaries are characterized by their singular behaviour and states that are close in the phase space are similar, except at the phase boundary. Therefore, the fidelity drops to 0 at a phase transition. With our approach of anomaly detection, we reinterpret the singular behaviour as a data anomaly. Moreover, as we explain next, we do it from scalable data. Our method does not require a full description of the physical states and therefore full state contraction can be avoided. Thus, for higher dimensional systems, [136] could not be used as a computational method as contraction is known to be generally inefficient for 2d tensor network states (commonly referred to as PEPS [137]).

In principle, there are many possible choices as input data for training our method, including the full state vector. To improve scalability and reach large system sizes, we propose to use quantities that arise naturally in the state description and do not require complete state information. For instance, we obtain ground states with tensor networks, from which we use the tensors themselves or the entanglement spectrum (ES) as input data. These quantities arise naturally from the state description without further processing and contain crucial information about the phase, like the ES for example [138, 139, 140]. We stress, however, that the choice of preferred quantities to be used for ML may in general vary and depend on the simulation method. In fact, we show that our method also works well with physical data accessible in experiments such as low-order correlation functions.

As a benchmark, we apply our method to the extended Bose Hubbard model in one dimension at exact integer filling. Its phase diagram is very rich and therefore provides a very good test to showcase our method. We are able to determine the entire phase diagram in a completely unsupervised and automated fashion. Importantly, our results point out the existence of a supersolid state that appears in the system in addition to the standard superfluid, Mott insulator, Haldane-insulating, and density-wave phases.

4.1 Methods

In this section we describe the ML technique and the physical model that we investigate in this chapter.

Anomaly Detection Method

We apply deep NN autoencoders (AEs) for anomaly detection [48]. As described in 2.1, an AE is a type of unsupervised NN that map the input x to a lower dimensional space $\boldsymbol{z}$ and map it back to $\bar{\boldsymbol{x}}$. The aim of the training is the minimization of a loss function $l(\boldsymbol{x}, \bar{\boldsymbol{x}})$ that measures the dissimilarity of the input x and the output $\bar{\boldsymbol{x}}$. Heuristically, we find that the *mean-square error* $l(\boldsymbol{x}, \bar{\boldsymbol{x}}) = |\boldsymbol{x} - \bar{\boldsymbol{x}}|^2$ suffices for our purposes and provides good results.

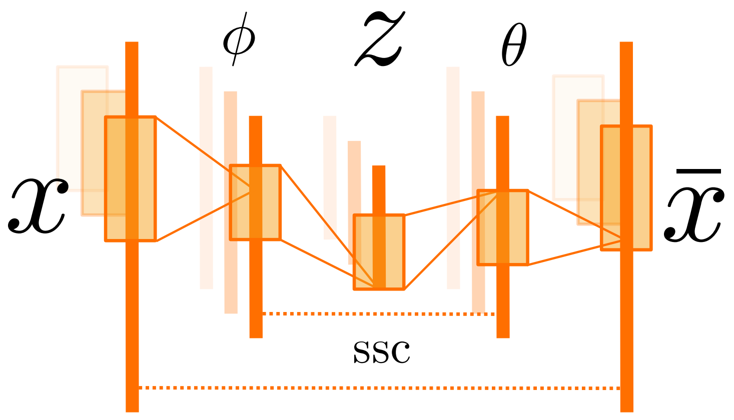


Figure 4.1: Schematic one-dimensional CNN AE with symmetric shortcut connections (SSC) that connect latent layers of the same dimension directly via addition, thereby improving the model performance [141, 142]. Shaded boxes in the background indicate schematically the convolutional nature of the layers. We illustrate how the input data $\boldsymbol{x}$ gets encoded into the lower dimensional latent vector $\boldsymbol{z}$ and decoded again to $\bar{\boldsymbol{x}}$.

The idea of this anomaly detection scheme is that for each state $|\psi\rangle$ we take corresponding data $\boldsymbol{x}$, such as for instance its ES or low order correlation functions. The AE learns the characteristic features of this data and encodes it into the latent variable z at the bottleneck [143], from which it is ideally able to reconstruct the original input. The loss l, that typical reaches $< 5\%$, directly indicates the success of this endeavour. We improve the performance of the AE by employing symmetric shortcut connections (SSC, see Figure 4.1). These SSC are direct connections between the layers of the encoder and the decoder and it has been shown that they improve the performance of AEs [141, 142]. After a successful training, the intuition is that, when confronted with data from unknown phases, the AE is unable to encode and decode $\boldsymbol{x}$. This leads to a higher loss, from which we deduce that the states do not belong to the same phase as the ones used to train the AE.

Deep learning architectures are known to generalize well [134, 135], such that it suffices to train in a small region of the parameter space. A mayor advantage of our method over known supervised deep learning methods is that the anomaly detection scheme does not rely on labeled data. We choose training data from one or several regions of the phase diagram, and ask how the loss of a test data point from any region of the phase diagram compares to the loss of these training points. As we show later, this can be performed with no a priori knowledge and in a completely unsupervised manner. The computationally most expensive step is the training and with our method it has to be performed only once to map the whole phase diagram, as opposed to multiple trainings like in [11, 15].

The specific architecture in use consists of two 1d-convolutional encoding and decoding layers with SSCs (Figure 4.1), implemented in TensorFlow [144]. To ensure the reproducibility of our results, we made the source code available under an open source license [145].

The extended Bose Hubbard Model

We test our method on the extended Bose-Hubbard Model in 1D with integer filling. The Bose-Hubbard (BH) model describes spinless bosons on a lattice that interact if they occupy the same site. The extended BH model also includes interactions of bosons that occupy neighbouring sites.

$$H = -t \sum_i \left(b_i^\dagger b_{i+1} + b_{i+1}^\dagger b_i \right) + \frac{U}{2} \sum_i n_i (n_i - 1) + V \sum_i n_i n_{i+1}, \tag{4.1}$$

This model serves as a highly non-trivial test ground with its rich phase diagram that, beside a critical superfluid and two insulating phases, admits a symmetry protected topologically ordered phase at commensurate fillings [146, 147, 148, 149, 150, 151, 152, 153, 138, 154]. Here, $n_i = b_i^\dagger b_i$ is the number operator for Bosons defined by $[b_i, b_j^\dagger] = \delta_{ij}$. Typically, we are interested in varying the on-site interaction U and nearest-neighbour interaction V and fix the hopping term $t = 1$. We explicitly enforce filling $\bar{n} := \sum_i \langle n_i \rangle / L_\infty = 1$ by employing $U(1)$ symmetric tensors [155], which we implement using the open source library TeNPy [156] (easily readable code accessible in [145]).

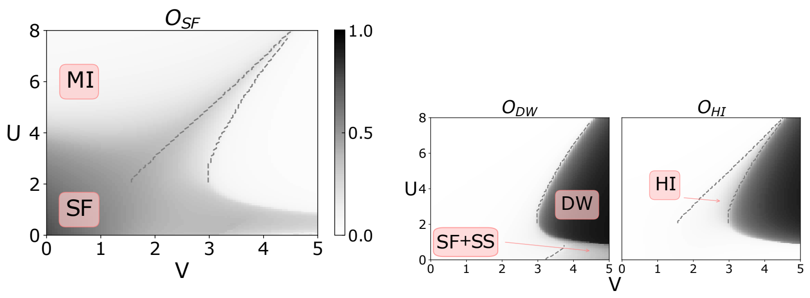


Figure 4.2: Extended BH phase diagram with five distinct phases obtained by the correlators in Equations 4.2 to 4.4. MI: Mott Insulator, SF: Super Fluid, SS: Super Solid, DW: Density Wave, HI: Haldane Insulator. The dashed lines indicate the transition points observed from diverging correlation lengths between MI-HI-DW and non-zero $\mathcal{S}$ in 4.10 between SF and SF+SS.

Figure 4.2 shows the phase diagram of the extended BH model with the Mott-insulating (MI), the superfluid (SF), density-wave (DW) and Haldane-insulating (HI) phases. Furthermore, we indicate a region where we found the SF phase coexisting with a supersolid (SS) phase. One way to physically classify these phases is to look at the correlators

$$C_{\text{SF}}(i,j) = \langle b_i^\dagger b_j \rangle \tag{4.2}$$

$$C_{\text{DW}}(i,j) = \langle \delta n_i (-1)^{|i-j|} \delta n_j \rangle \tag{4.3}$$

$$C_{\text{HI}}(i,j) = \langle \delta n_i \exp\left(-i\pi \sum_{i \le l \le j-1} \delta n_l \right) \delta n_j \rangle \tag{4.4}$$

from which we will construct the order parameters, with $\delta n_i = n_i - \bar{n}$, which measures the difference of the occupation of the i-th site and the average filling. The DW phase occurs in a region of strong nearest neighbour repulsion and therefore the bosons start arranging in patterns where one site is occupied and the neighbouring site is empty. This pattern is best captured with C_{DW} that converges to a constant value in the DW phase. The HI phase experiences topological order and long-range density-density correlation functions will decay exponentially. Nevertheless, there is some hidden long-range order in the HI phase that is expressed with the string order parameter C_{HI} which calculates the correlation function between site i and j while accounting for all the sites in between i and j. Hence, while the single sites with distance $|i-j|$ might not show a correlation, taking all the sites in between i and j into account there is a correlation [157]. This can be illustrated by a state, where δn_i shows a patter $[+1, 0, 0, -1, 0, +1, -1, 0, 0, +1, \ldots]$. Such a pattern does not have an obvious long range correlation $\langle \delta n_i \delta n_j \rangle$, but if we ignore the sites with $\delta n_i = 0$ we see that the occupation number switches between $+1$ and -1 and the order parameter $C_{\mathrm{HI}}(i,j)$ takes exactly this into account.

In the superfluid phase the long range off-diagonal order C_{SF} discriminates the Mott-insulating (MI) phase and the superfluid (SF) phase, where it decays with a power-law in the SF phase and exponentially in MI. More details about the characterization of the system can be found in [146] and the Appendix C. We visualize the phase diagram by computing the order parameter from the correlators above $O_\bullet = \sum_{i,j} C_\bullet(i,j)/L_\infty^2$ in 4.2 in the thermodynamic limit for a repeating unit cell of $L_\infty = 64$ sites with a maximum bond dimension $\chi_{\max} = 100$ and assuming a maximum occupation number $n_{\max} = 3$, which results in a local dimension $d = n_{\max} + 1 = 4$. We use data from these states, obtained with these parameters throughout the rest of the following analysis.

Simulation Method and Input Data

We calculate the ground states by means of the Density Matrix Renormalization Group algorithm (DMRG) in terms of Tensor Networks, i.e. Matrix Product States (MPS) [158, 137]. A general multipartite state of L parties with local dimension d $|\Psi\rangle = \sum_{\boldsymbol{\sigma}} c_{\boldsymbol{\sigma}} |\boldsymbol{\sigma}\rangle$, where $\boldsymbol{\sigma} = \sigma_1 \ldots \sigma_L$ is the vector of local indices $\sigma_i = 1, \ldots, d$, can always be decomposed into products of tensors with the aid of the singular value decomposition that has the left-canonical form

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} A^{\sigma_1} \cdots A^{\sigma_L} |\boldsymbol{\sigma}\rangle\,, \tag{4.5}$$

or the mixed-canonical form,

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} A^{\sigma_1} \cdots A^{\sigma_{i-1}} \Theta^{\sigma_i} B^{\sigma_{i+1}} \cdots B^{\sigma_L}, \tag{4.6}$$

where $\{A^{\sigma_j}\}_{j=1}^{i-1}$ and $\{B^{\sigma_j}\}_{j=i+1}^{L}$ are left- and right-normalized, which is (see [158] for details)

$$\sum_{\sigma_j} A^{\sigma_j}\left(A^{\sigma_j}\right)^{\dagger} = \sum_{\sigma_j}\left(B^{\sigma_j}\right)^{\dagger} B^{\sigma_j} = \mathbb{I}. \tag{4.7}$$

The tensor Θ^{σ_i} is left over after all of the sites in the MPS have been orthonormalized and we will refer to it as the central matrix. It is also referd to as the single site pseudo wavefunction and can be used to calculate local observables.

For our work we use the convention of Vidal [159], and write our ground state in the MPS form

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \Gamma^{\sigma_1}\Lambda^{[1]}\cdots\Lambda^{[i-1]}\Gamma^{\sigma_i}\Lambda^{[i]}\cdots\Lambda^{[L-1]}\Gamma^{\sigma_L}\left|\sigma_1\ldots\sigma_i\ldots\sigma_L\right\rangle, \tag{4.8}$$

where $\Lambda^{i-1}\Gamma^{\sigma_i} = A^{\sigma_i}$ and $\Gamma^{\sigma_i}\Lambda^{i} = B^{\sigma_i}$. This representation has the advantage that $\Lambda^{[i]}$ is the diagonal singular value matrix of a bipartition of the chain between site i and $i+1$, and contains the Schmidt values (see [158]). The approximates of the exact ground state can be done by keeping only the $\chi_{\max}$ largest Schmidt values for each partition, where $\chi_{\max}$ is known as the bond dimension. This is the best approximation of the full state in terms of the Frobenius norm and enables us to handle big system sizes. Eq. (4.8) corresponds to finite length and open boundary conditions. Here, we use the version formulated in the thermodynamic limit for infinite MPS (iMPS) [160, 161, 162]. In this case, instead of a finite chain, we are effectively operating in the thermodynamic limit and have a finite but repeating unit cell of length L_∞.

There is a natural graphical language for tensor networks, where Equation 4.8 corresponds to Figure 4.3 [158]. In this graphical representation, connected lines correspond to index contractions, which in the case of Eq. 4.8 corresponds to matrix multiplication. As aforementioned, in order to calculate local observables at site i, one only needs the *single site pseudo-wavefunction*

$$\Theta^{\sigma_i}_{v_{i-1},v_i} = \sum_{a,b} \Lambda^{[i-1]}_{v_{i-1},a}\Gamma^{\sigma_i}_{a,b}\Lambda^{[i]}_{b,v_i}. \tag{4.9}$$

For example, single-site operator expectation values are simply calculated in terms of $\langle O_i\rangle = \sum_{\sigma_i,\sigma_i',v_{i-1},v_i} \Theta^{\sigma_i}_{v_{i-1},v_i} O_{\sigma_i,\sigma_i'}\left(\Theta^{\sigma_i'}_{v_{i-1},v_i}\right)^*$

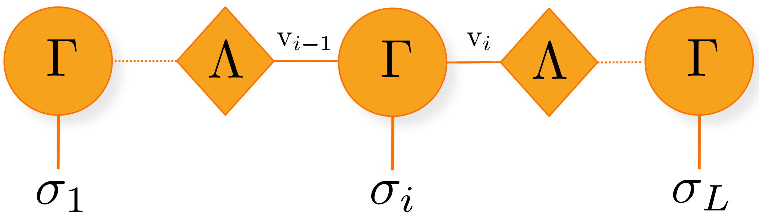


Figure 4.3: Graphical representation of a finite MPS with open boundary conditions, physical indices σ_i and virtual indices v_i. $\Gamma^{\sigma_i}_{v_{i-1},v_i}$ is the local description of site i with singular values Λ connecting to the left and right part of the chain (entanglement spectrum).

and similarly two-point correlation functions, as graphically depicted in 4.4.

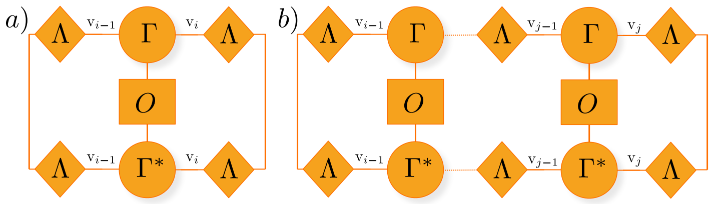


Figure 4.4: Graphical representation of a) single-site expectation value $\langle O \rangle$ and b) two-point correlators $\langle O_i O_j \rangle$.

For the training of the NNs we use different quantities as input data $\boldsymbol{x}$ and to explore the phase diagram. First we use the Schmidt values $\Lambda^{[i]}$ as an input and we ambiguously refer to it as ES. Our numerical results support the functionality of using this anomaly detection scheme with ES as we get near-constant losses for states of the training region and significantly higher losses for unknown phases. The method generalizes well with similar losses for states inside and outside the training region. For the ES as an input we only need one dimensional convolutional layers for the feature extraction. The tensor $\Theta^{[i]\sigma_i}_{v_{i-1},v_i}$ has three indices [158] and to input it into a NN, we treat the tensor as an image, that, instead of the typical red green blue (RGB) channels, has “spin channels". Each excitation value $\sigma_i = 0, \cdots, n_{\max}$ now corresponds to a “colour channel“ and the two indices v_{i-1} and v_i can be interpreted as the two dimensions of the image, yielding a $\chi_{\max} \times \chi_{\max}$ image for that channel. This way, we can use classical ML methods that are optimized for image processing for our physical problem, namely 2D CNNs.

4.2 Results

This section is dedicated to the results that we obtained with our anomaly detection scheme applied to the extended Bose Hubbard model.

Numerical Results

Assuming no a priori knowledge, we start by training with data points at the origin of the parameter space $(U, V) \in [0, 1.3]^2$ indicated as a blue window in Figure 4.5, which accounts to training in SF. By testing with data points from the whole phase diagram we can clearly see the boundaries to all other phases from SF in Figure 4.5. The transition between SF and MI is matched by an abrupt rise in loss (Figure 4.5, inset a)). In this particular case, we can already determine the different phases inside the anomalous region due to their different loss levels and the appearance of two valleys at the phase boundaries between MI, HI and SF (Figure 4.5, inset b)). Physically, we can explain these valleys by the criticality of these Luttinger and Ising type transitions, which lead to a slowly decaying ES at the boundary, just like in the critical SF phase.

It is not necessarily always the case that one can differentiate the different phases inside the high-loss anomalous region after solely training in one single region. Thus, as a systematic approach, we propose picking homogeneous and high contrast anomalous regions after the initial training and retrain the AE with training inputs coming from these regions. Here, we were somewhat fortunate to start our training only in the SF phase and we could already map out the whole phase diagram after the first training iteration. It could be that one picks accidentally to train in two phases at the same time. Therefore, we suggest to redo the training for different training sets. In the Appendix B we show what happens if we train in the region $(U, V) \in [4, 4.8] \times [2, 4]$, which was a high loss region for the training in Figure 4.5.

Our method is not tailored to ES as input data. To show this, we use on one hand the central tensors Θ from the MPS as

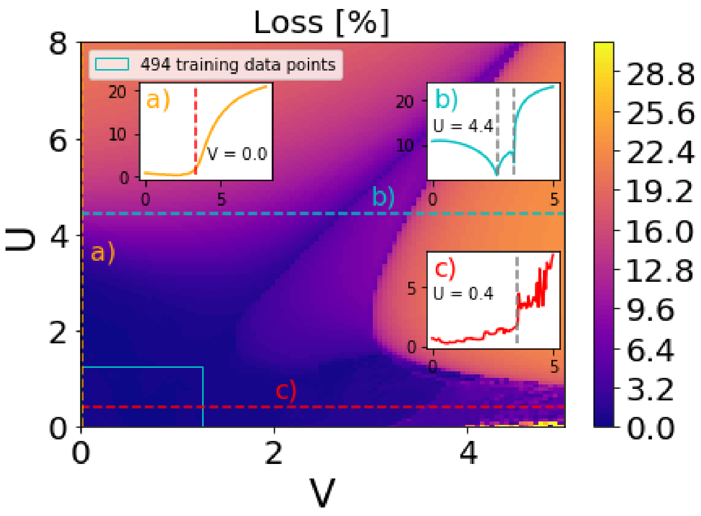


Figure 4.5: 2D loss map of the AE after training near the parameter space origin (blue square frame). The insets a), b) and c) show the loss along the dashed lines. Vertical green dashed line in inset a) indicates critical $U_c = 3.33$ [148]. Vertical grey dashed lines in inset b) and c) are the transitions from 4.2. The phase boundaries are determined by a rise in loss (inset a) and c)). The anomalous regions are already well-separated by decreasing losses because of the critical behaviour at the phase boundaries (inset b)), which share similarities with the critical SF phase. Higher loss indicates that this region is more different from the training region in the blue square, lower loss indicates similarity.

input data in Figure 4.6. On the other hand we use experimentally accessible correlators in Figure 4.8. We see that the method works well when using the central tensor as input data. Furthermore, we find that the network is capable of encoding more than one phase in the training dataset. We still find the boundaries between MI, HI and DW due to the criticality of the transitions (see 4.6, inset a)), similar to the valleys in Fig. 4.5. Even though translational invariance is broken in DW, we find it suffices to use only one tensor Θ from the center of the unit cell. This is because, despite the broken translational symmetry, entanglement is still distributed uniformly in the unit cell, which is implicitly encoded in Θ. Heuristically, we found that already a tensor or the ES from a site is sufficient for our method to discriminate the phases. A possible explanation is the translational invariance of

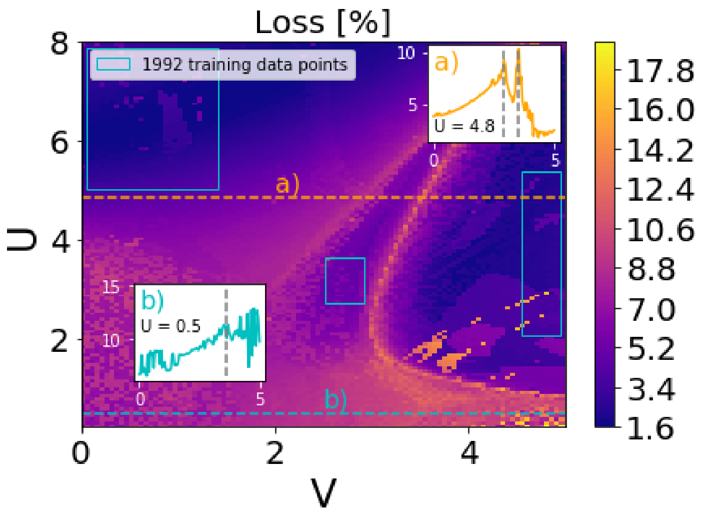


Figure 4.6: Instead of the ES, we use the central tensor Θ as input data for the AE and use 2D convolutional layers. The same AE can encode both MI, HI and DW data.

the model, which, in our case also results in uniformly distributed entanglement. By this, we mean that the entanglement spectrum is the same at all bonds, resulting in a uniform entanglement entropy distribution along the chain. This is true even for the DW phase, where translational invariance is broken. Only for the SS phase (discussed in more details below) this is not the case anymore, where the ES admits a periodic pattern. To improve the results one could take several spectra or tensors from the chain by concatenating them, keeping the input data scalable. Yet, we find that one tensor or ES suffices for discrimination.

For the use of correlators as an input, instead of using unprocessed data from simulation, we calculate $\{C_{\text{SF}}(i,j)\}_{i,j=1}^{64}$ and train in MI and SF simultaneously. We interpret the rows of the matrix C_{SF} as color channels for a 1D CNN. Because C_{SF} does not contain any information about the topological order in HI, the method does not recognize this region as we would expect (4.8, inset a)). Overall, the boundaries match perfectly with a sharp increase onto a plateau at the transition points. This opens the possibility to use physical observables from experiment with the caveat of requiring physical knowledge a priori.

By close inspection of 4.8, we see a region with noticeable contrast for small U and large V, indicating the presence of a separate phase. This is interesting because, initially, we did not expect to find a fifth phase in the diagram. Upon further physical investigation, we find a phase-separated state between SF and supersolid (SF + SS). Supersolidity in this model has been studied in previous literature for incommensurate fillings [147, 148, 149, 163] and was claimed to be found for filling 1 in [138] without further discussion. The phase separation that we find here is new and has not been studied before to the best of our knowledge. In order to physically show the transition, we compute the Fourier transform of the local density $\tilde{n}(k) = \sum_j \langle n_j \rangle e^{-ikj}/L_\infty$ and detect long-range solid order by looking at

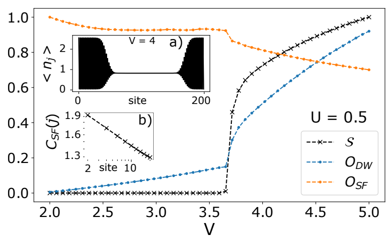
$$\mathcal{S} := \max_{k \neq 0} |\tilde{n}(k)|^2 \tag{4.10}$$

in 4.7(a) [164]. Additionally, we find non-zero O_{DW} and O_{SF}, showing both superfluid and crystalline behavior. For higher numerical accuracy and better illustration of the correlator decay, we compute a larger state for $L_\infty = 200$, $d = 6$ and $\chi_{\max} = 500$ at $(U, V) = (0.5, 4)$ and see both crystalline and superfluid regions in the density profile $\langle n_j \rangle$, 4.7(a) inset a). To confirm supersolidity of the crystalline part we show that C_{SF} decays with a power-law in that region, see 4.7(a) inset b).

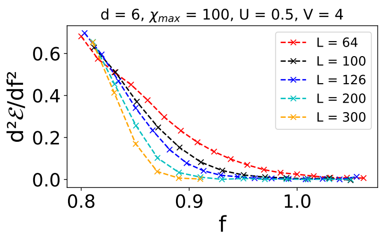
This phase separation occurs as the system becomes mechanically unstable. We can see this as the second derivative of the ground state energy per site $\mathcal{E} = E/L$ with respect to the filling f vanishes. We perform finite size scaling with open boundary conditions to show this in 4.7(b). There, we target equidistant discrete fillings $f_i = N_i/L$ for $N_i \in [0.8L, 1.1L]$ and compute the finite difference derivative

$$d^2\mathcal{E}/df^2 = \left(\mathcal{E}(f_{i-1}) - 2\mathcal{E}(f_i) + \mathcal{E}(f_{i+1})\right)/(f_i - f_{i-1})^2. \tag{4.11}$$

The detection of this new phase demonstrates the power of our approach and we discuss further findings in the following section. An extensive study of the SS-SF phase separation we leave to future work.



(a)



(b)

Figure 4.7: a) Transition from SF to phase separated SF + SS at fixed U=0.5. The solid long-range order emerges while SF correlations sustain. Inset a) shows the phase separation in the density $\langle n_j \rangle$ for a state at $(U, V) = (0.5, 4)$ with $L_\infty = 200$, $d = 6$ and $\chi_{\max} = 500$. Inset b) shows the power-law decay of $C_{\text{SF}}(0, j)$ in the solid part via doubly logarithmic plot of every second value, confirming supersolidity. b) Finite size scaling of the vanishing second derivative of the ground state energy per site $\mathcal{E}$ with respect to the filling f. This shows that the system becomes mechanically unstable, leading to phase separation as depicted in a) for $f = 1$.

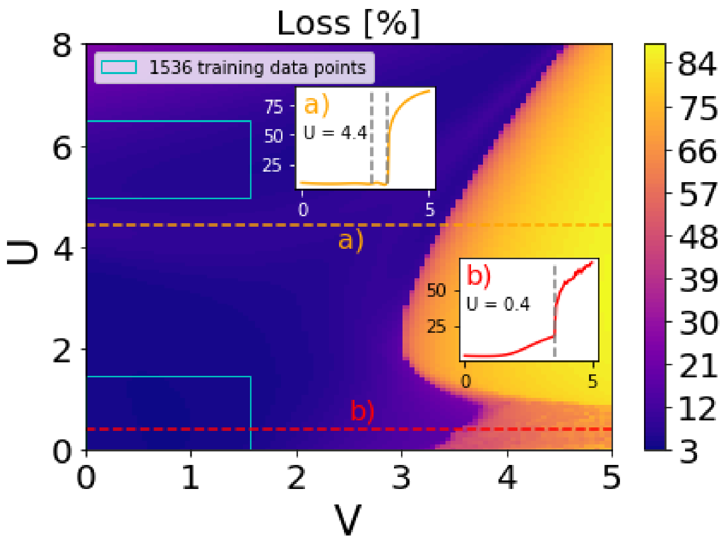


Figure 4.8: 2D loss map of the AE after training in the two blue square frames in the SF and the MI phase. The insets a) and b) show the loss along the dashed lines. Instead of the ES, we use the physically accessible correlator C_{SF} as input data. The HI is not recognized as this correlator does not contain information about the topological order of this phase.

Phase separated supersolid and superfluid

In this section we give more details about the discovered phase separated ground states between supersolid (SS) and superfluid (SF) for large V and small U at filling 1. First, we note that iDMRG does not converge for small unit-cell sizes, as exemplarily shown for $(U, V) = (0.5, 4)$ and $L_\infty \in \{2, 4, 8, 16, 32, 64\}$ in 4.9. The change of energy is checked every 25 sweeps, where we set the threshold for convergence to 10^{-8}. We find that iDMRG does not reach convergence after 1000 sweeps for all runs but $L_\infty = 64$. There, the system has enough space to phase-separate into the two phases and iDMRG can converge. One convincing argument that this is a physical effect and not a numerical artefact is that we obtain better (smaller) ground state energies per site E_0/L_∞ as we increase the unit cell-size, as depicted in 4.10. We can see a breaking point in computation time at $L_\infty = 64$ as the system starts to converge to a solution. Choosing $L_\infty = 64$ for the simulations in the main text is a trade off between computation time and accuracy in energy. Another argument to convince ourselves of this effect was to increase the hyper-parameters d, χ_{max} and L_∞. The ground states we obtained showed the same properties, that we are now describing for $L_\infty = 200$, $d = 6$ and $\chi_{\text{max}} = 500$ at $(U, V) = (0.5, 4)$: We show the correlator C_{SF} with respect to a site inside the solid part in 4.11a). In terms of the density (4.11a) inset 2), the system has a solid part with alternating filling $\langle n_j \rangle \approx (\cdots, 2.5, 0, 2.5, 0, \cdots)$, and a superfluid part with constant filling $\langle n_j \rangle < 1$. We see that inside the solid part, C_{SF} decays with a power law, making it SS (4.11a) inset 1). This is new and different to the supersolids that have been discussed at incommensurate fillings in previous studies [147, 148, 149, 138, 163]. We can recover such a homogeneous supersolid by increasing the filling, see 4.11b). Here, we find a homogeneous density (inset 2), accompanied by power law decaying C_{SF} (inset 1), as is typical for SS.

Note that in the case of phase separation the translational invariance is broken and therefore iDMRG is not optimal. However, we find matching results when repeating the calculations for finite chains with open boundary conditions (OBC). These states with OBCs are used to show that the system becomes mechanically unstable around filling 1 in the main text in Fig. 4.7(b).

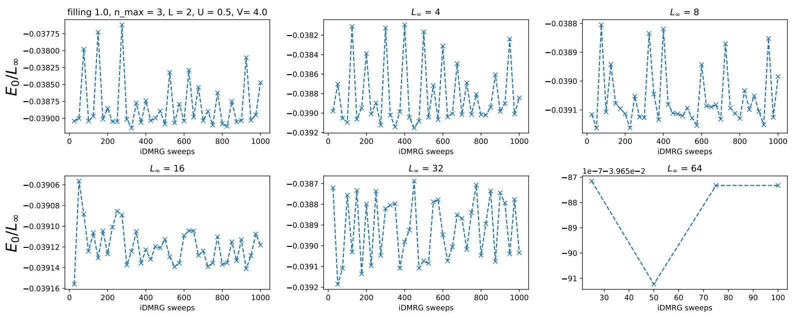


Figure 4.9: Ground state energy during iDMRG iterations for $L_\infty = 2, 4, 8, 16, 32,$ and 64. Convergence criteria are checked every 25 sweeps. The maximum number of sweeps was set to 1000 which was reached without convergence for all but the $L_\infty = 64$ run.

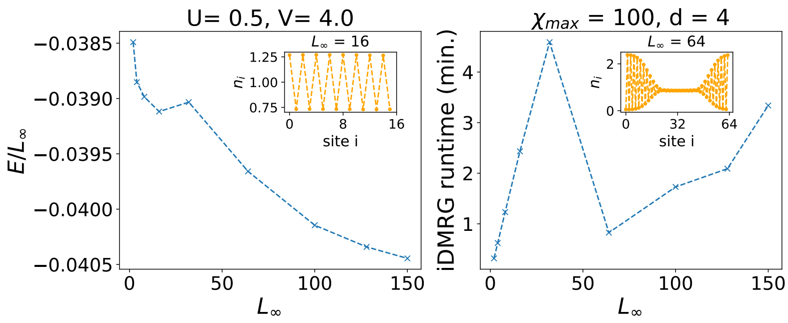
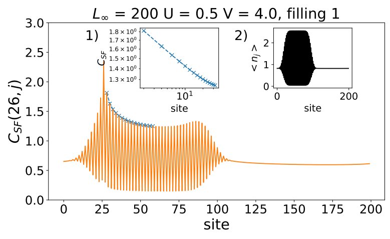
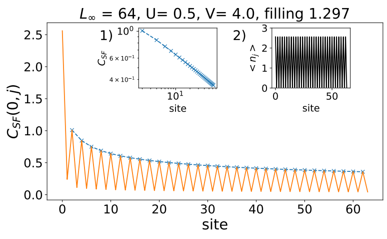


Figure 4.10: Energy per site (left) and computation time (right) for unit cell sizes $L_\infty = 2, 4, 8, 16, 32, 64, 100, 128$ and 150 with iDMRG. For $L_\infty = 64$ and upward the system has enough space to phase separate and iDMRG starts to converge, leading to a breaking point in computation time. The insets show the density profile $\langle n_i \rangle$ exemplarily for $L_\infty = 16$ and 64.



(a)



(b)

Figure 4.11: a) $C_{\mathrm{SF}}(i, j)$ for the phase-separated ground state at filling 1 for $L_\infty = 200$, $d = 6$ and $\chi_{\max} = 500$, where i is taken within the solid region (here site 26). The doubly logarithmic plot in inset 1 reveals the power law decay of C_{SF} inside the solid region, confirming supersolidity. Inset 2 shows the density $\langle n_j \rangle$ along the chain with solid checker board and flat superfluid parts. b) $C_{\mathrm{SF}}(i, j)$ for a homogeneous supersolid ground state at higher filling 1.297. Now the whole chain has a solid checker board pattern (inset 2) while C_{SF} decays with a power law (inset 1), making the ground state overall supersolid.

4.3 Conclusion

We have shown an unsupervised method to map out the phase diagram of a complex quantum many-body system that could possibly be performed fully data driven and without physical apriori knowledge such as the construction of an order parameter. By using tensor networks we can reliably compute ground states of many-body systems in the thermodynamic limit and at the same time extract the desired data without further processing. Entanglement spectra and central tensors serve as natural quantities in this context, but the method also proved successful for physical observables like $\langle b_i^\dagger b_j \rangle$ correlators or could also be used for the unprocessed wavefunction as an input. Hence, this method can be applied in both purely computational platforms like self-driving laboratories as well as experimental setups. It is as well worth stressing that our method indeed helped us to discover a region of interest in a well studied model like the extended BH model. The region at low U and high V shows phase separation between a SF and an SS phase and we would not have discovered this without our anomaly detection method. An in-depth study of this region of the phase diagram is currently being prepared but is not yet available as a preprint.

Chapter 5

Interpretability of NN phase prediction

As discussed in previous chapters, deep fully connected and convolutional NNs have been applied to detect phase transitions in a variety of physical models. Next to all these successful applications, there are still some open problems. For instance, concerning topological models and many-body localization (MBL), many methods rely on pre-engineered features [165, 166, 167], they show high sensitivity to hyperparameters describing the training process [18] or the critical exponent is still in disagreement with predicted values as we described in Chapter 3. Moreover, even in the models described by Landau's theory, so far, these approaches have mostly enabled only the recovery of known phase diagrams or the location of phase transitions in qualitative agreement with more conventional approaches based, for instance, on order parameters and/or theory of finite-size scaling. Nonetheless, ML achieved this at a much lower computational cost, e.g. using fewer samples as described in Chapter 3 or smaller system sizes [18].

Additionally, the used ML techniques are mostly black boxes, i.e., systems with internal logic not obvious at all to a user [168]. The missing key element is the model interpretability, i.e., the ability to be explained or presented to a human in understandable terms [169]. The research on this crucial property is at the heart of a booming field of ML interpretability [170, 171, 172, 173, 174, 175] aiming at designing methods that discover the internal logic of commonly used black boxes. They are needed for a plethora of reasons. Given the ML presence in everyday life, it is no surprise that already legal measures have been taken to assure that any individual can obtain meaningful explanations of the logic involved when automated decision-making takes place [176].

Next to the legal motivations, there is ethics. The worrying fact was revealed that learning machines inherit biases from humans preparing data [177]. Also, deep NNs were shown to perfectly fit random labels [178], and that a group of local features can be their good approximation [179]. These studies prove that the learning process sometimes goes against our intuition, and indicate that the predictions should be accompanied by a justification understandable by humans to be trusted. Detecting phases of matter with ML methods is no exception from this and the desire to know and understand the underlying mechanisms of the process cannot be met.

This need for interpretation has been already stressed by physicists, but proposed methods are either restricted to linear models [24] or the particular NN's architecture [19], or require pre-engineering of the data, and, as a result, they are very specific to both the ML and physical model [49]. Hence, in this work, we follow a different paradigm without relying on the *a priori* knowledge of the order parameter or the system itself. Our method is straightforwardly applicable to any physical model or experimental data with no dependence on the architecture of the ML model. We show how interpretability methods can be used in the classification of quantum phase transitions to understand what characteristics are learned by a ML algorithm. This universal approach unravels if a relevant physical concept was indeed learned or if the prediction cannot be trusted. We also present that an interpretable NN can give additional information on the phase transitions, not provided to the algorithm explicitly.

5.1 Methods

As described in 2.1, we consider supervised learning. In this Chapter we remove training points with their labels from the training data. Therefore, we use the short hand notation for the labeled training data $\mathcal{D} = \{z_i\}_{i=0}^{n}$, with $z_i = (\boldsymbol{x}_i, y_i)$. In our setup, the inputs $\boldsymbol{x}_i$ are the state vectors for a given physical system, and y_i are the corresponding phase labels. The ML model is determined by the set of parameters $\boldsymbol{\theta}$. In the training process, the parameters' space is being searched for the final parameters $\hat{\boldsymbol{\theta}}_{\mathcal{D}} \equiv \hat{\boldsymbol{\theta}}$ of the ML model, which minimizes the empirical risk $\mathfrak{L}(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{z \in \mathcal{D}} \mathcal{L}(z, \boldsymbol{\theta})$, where n is the training data set size, which tends to be of the order of thousands. After training, a

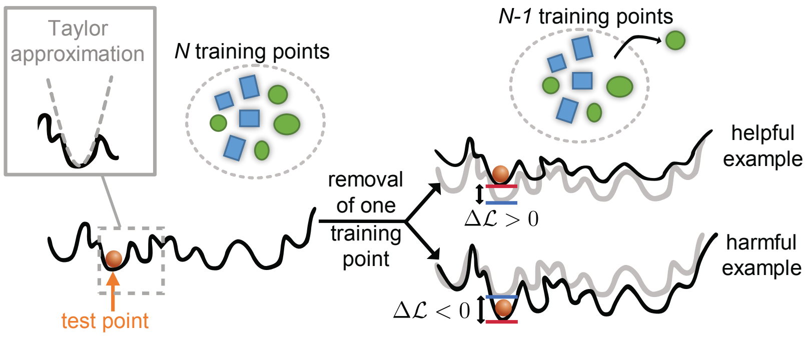


Figure 5.1: (a) Visual explanation of leave-one-out training and its approximation, the influence function. The black solid line represents the loss landscape of the NN with its local minima. The removal of a training point can lead to an increase or a decrease in loss.

model can make a prediction for an unseen test point z_{test} with the test loss function value $\mathcal{L}(z_{\text{test}}, \hat{\boldsymbol{\theta}})$ related to the model certainty of this prediction.

Interpreting neural networks with influence functions

In Section 2.1 we introduced the idea of interpretable ML methods, its importance and how it can help to understand why for example a NN comes to a certain conclusion. In this chapter we study an intuitive way of unraveling the logic learned by the machine. The idea is to retrain the model after removing a single training point z_{r} (starting from the same minimum, if a non-convex problem is analyzed), and checking how it changes the prediction of a specific test point z_{test}. Such a leave-one-out training (LOO) [73] studies the change of the parameters $\boldsymbol{\theta}$, now shifted to a new minimum $\hat{\boldsymbol{\theta}}_{\mathcal{D}\setminus\{z_{\text{r}}\}}$ of the loss function, as depicted in Fig. 5.1. An analysis of the test loss change, $\Delta\mathcal{L} \equiv \mathcal{L}(z_{\text{test}}, \hat{\boldsymbol{\theta}}) - \mathcal{L}(z_{\text{test}}, \hat{\boldsymbol{\theta}}_{\mathcal{D}\setminus\{z_{\text{r}}\}})$, enables the indication of the most influential training points for a given test point z_{test} being the ones whose removal causes the largest change. Influential examples can be both helpful ($\Delta\mathcal{L} > 0$) and harmful ($\Delta\mathcal{L} < 0$). Such an analysis gives the notion of a similarity used by the machine in a given problem, as training points being the closest in the $\Delta\mathcal{L}$ space can be understood as the most similar. Once the most influential points are identified, we can decode what characteristics are being looked at by comparing ‘similar’ points in the machine 'understanding'. It can be especially useful in phase classification problems where the analysis of $\Delta\mathcal{L}$ enables the recovery of patterns being crucial for distinguishing the phases. Therefore, we can gain some understanding of what a ML algorithm is learning if we use a supervised setting for phase discrimination and if it learns something that makes physically sense, like an order parameter. The use of this technique to check the influence of every training point in $\mathcal{D}$ on a given test point is, however, prohibitively expensive, as the model has to be retrained for each removed z.

To circumvent this problem, one can make a Taylor expansion of the loss function $\mathcal{L}$ with respect to the parameters around the minimum $\hat{\boldsymbol{\theta}}$, and approximate $\Delta\mathcal{L}$ resulting from the LOO training. This method was proposed for regression problems already forty years ago [73, 74, 75] and named influence functions. Not only this interpretability method is computationally feasible, but also it treats a model as a function of the training data instead of assuming that the training data set is fixed. The influence function reads

$$\mathcal{I}(z_{\mathrm{r}}, z_{\mathrm{test}}) = \frac{1}{n}\nabla_{\boldsymbol{\theta}}\mathcal{L}(z_{\mathrm{test}}, \hat{\boldsymbol{\theta}})^T H_{\boldsymbol{\theta}}^{-1}(\hat{\boldsymbol{\theta}})\nabla_{\boldsymbol{\theta}}\mathcal{L}(z_{\mathrm{r}}, \hat{\boldsymbol{\theta}})\,, \tag{5.1}$$

and it estimates $\Delta\mathcal{L}$ for a chosen test point z_{test} after the removal of a chosen training point z_{r}. Here, $\nabla_\theta\mathcal{L}(z_{\mathrm{test}}, \hat{\boldsymbol{\theta}})$ is the gradient of the loss function of the single test point, $\nabla_\theta\mathcal{L}(z_{\mathrm{r}}, \hat{\boldsymbol{\theta}})$ is the gradient of the loss function of the single training point whose removal's impact is being approximated, and $H_{\boldsymbol{\theta}}^{-1}(\hat{\boldsymbol{\theta}})$ is the inverse of the Hessian,

$$H_{i,j}(\hat{\boldsymbol{\theta}}) = \frac{\partial^2}{\partial_{\theta_i}\partial_{\theta_j}}\mathcal{L}(\mathcal{D}, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}. \tag{5.2}$$

All derivatives are calculated w.r.t. the model parameters θ, evaluated at $\hat{\boldsymbol{\theta}}$ corresponding to the minimum of the empirical risk, $\mathcal{L}(\mathcal{D}, \hat{\boldsymbol{\theta}})$. We can only ensure the existence of the inverse of the Hessian if it is positive-definite. However, it was shown [76, 77] that this method could be generalized to non-convex problems and therefore applied to ML. The example code can be found in [180].

Influence function of Gaussian mixtures

To gain a better understanding of how influence functions work, we start with a simple classification example. We train a NN on two different datasets consisting of two-dimensional data points 5.2 drawn randomly from two different Gaussian distributions with mean $(1, 1)$ and $(-1, -1)$. The first dataset is depicted in Figure 5.2(a) and the second in Figure 5.2(b). We label the samples according to which Gaussian distribution they are coming from. The NN is a simple feed-forward architecture with 10 hidden units. The Figures 5.3(a) and 5.3(b) show the influence function of a NN that was trained on the data set with covariance 1.0 where the samples slightly overlap. The NN is not able to classify all the data points correctly, because in the overlapping region the data is indistinguishable. The NN is trained on this data until it converges and only a few points in the overlap region are missclassified. With the influence function we can now identify training points that are most influential for a certain test point. In Figure 5.3 we show the influence of the training points on a test point (marked with the black X). The color scheme indicates points with positive influence in red and negative influential points in blue. In the inset of both subfigures we show the influence of all points ordered by magnitude, to display the change in magnitude for different test points.

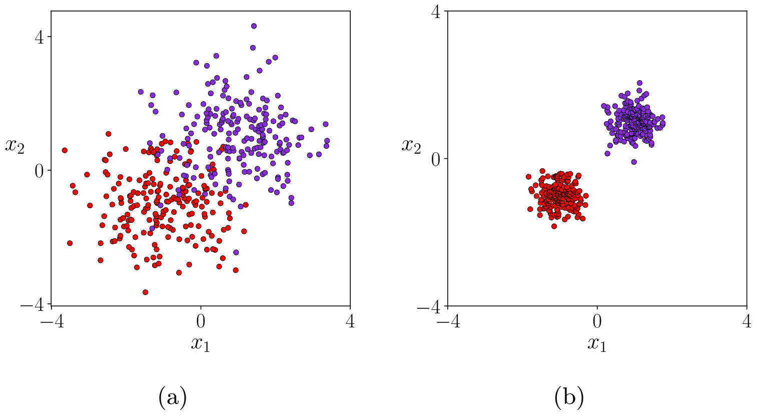


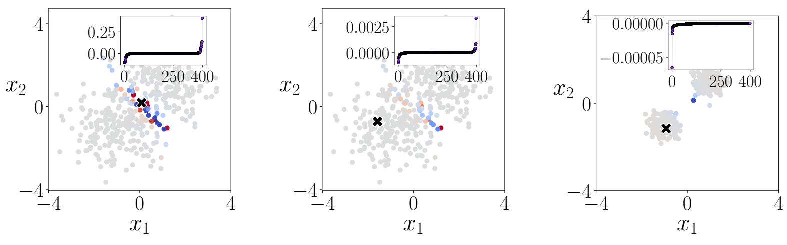
Figure 5.2: Gaussian mixture data: Data points drawn randomly from two Gaussian distributions with mean $(1, 1)$ and $(-1, -1)$ with Covariance a) 1.0 and b) 0.1

In Subfigure 5.3(a) we show a test point inside the overlapping region, where the NN is undecided, because it cannot distinguish from which Gaussian distribution the data point comes from and the test prediction is wrong. The test point is surrounded by several highly influential training points which means that removing them from the training set would change the prediction substantially. Subfigure 5.3(b) shows the same NN with the same parameters, but with a test point further inside one of the Gaussians. Again the most influential training points are close to the boundary between the distributions, but the order of magnitude of the influence function (inset of Figure 5.3(b)) is smaller than for the test point in the overlapping region. Finally, Subfigure 5.3(c) shows the influence of a NN trained on the dataset where the covariance of the Gaussians is 0.1 and the samples do not mix. One can see in the inset that there are no points with a positive influence and that a few points close to the decision boundary function as negative examples. This Figure shows that in such a clear decision task with no ambiguity, the NN loss cannot be influenced by removing a single training point.

In Chapter 6 we will discuss the information that is contained in the Hessian of a loss landscape more extensively and we study the Hessian for VQC loss landscapes. Generally, minima of NN in the parameter space are mostly flat pools that have just a few directions of steep ascent. In Equation 5.1 the inverse of the Hessian is used to adjust the overlap of the gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}(z_r, \hat{\boldsymbol{\theta}})$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}(z_{\text{test}}, \hat{\boldsymbol{\theta}})$ to the curvature of the loss landscape. The highest change in the loss results if these two gradients (anti)align and point in a flat direction of the loss landscape of the trained NN. In a direction of low curvature these gradients would lead to a substantial change in the parameters and therefore change the test loss the most. If the gradients would (anti)align in a direction of high curvature the change in the parameter space would be less pronounced. If the gradients do not strongly overlap or are even orthogonal, then the influence is zero.

Neural Network model for phase classification

For the phase classification we use a CNN (see Fig 5.4) consisting of 3 one-dimensional convolutional layers with 5 filters on the input vector, 8 filters on the first hidden layer and 10 filters for



(a) (b) (c)

Figure 5.3: Influence function for Gaussian mixtures: The figures show the influence of the training points on a test point (marked as a black X). The color scheme of the influence function is rescaled for visibility and does not refer to an explicit value, but helps to see the points with highest and lowest influence. The amplitude of the actual influence can be seen in the inset, where the influence of each point is shown ordered by magnitude. Blue points have a negative influence and therefore they are harmful for the prediction, red points have a positive influence and are therefore helpful for the prediction. a) Shows the influence of a test point inside the decision boundary, where the NN is maximally confused. There are many positively and negatively influential points close to the test point. b) This is the same NN as in a) with the same parameters, but with a test point deep inside one class. The influence is orders of magnitudes smaller and the NN prediction would not change as much if one of the points would be removed. c) The same NN trained on a dataset with non-overlapping Gaussian distributions. The influence is another two orders of magnitudes smaller and the point closest to the decision boundary is negatively influential and slightly confuses the NN.

the last convolution layer. After the first 2 convolutions we apply a max pooling layer to reduce the dimension, and the last convolutional layer is followed by a global average pooling (GAP) layer. The GAP architecture has been introduced in [181] and has found applications in discriminative localization of objects in image data [172]. It reduces each filter of the final convolution to a single value. After the GAP we have one fully connected layer with two output neurons that predict the labels. The GAP makes sure that most of the weights of the NN are contained in the convolutional part and, therefore, reduces the amount of weights in the fully connected part of the NN. This is important in our case, because we calculate the exact Hessian for the influence function which is computationally demanding for NN with

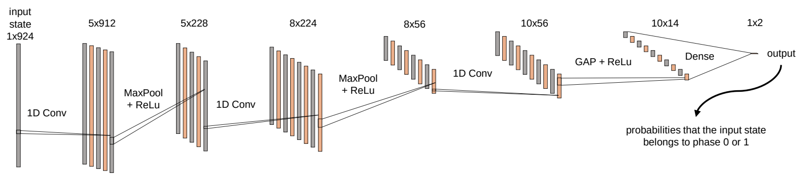


Figure 5.4: Scheme of used architecture. Length scale does not apply.

a lot of parameters. For such big NNs the Hessian can also be approximated like in [76] with a stochastic approximation. For the training of the NN we use state vectors from each phase as an input and label them with 0 or 1 for each phase. The state vectors are obtained via exact diagonalization of the Hamiltonian 5.3.

We use L_2 regularization during the training to effectively decrease the certainty of the NN's predictions. Actually, the undertrained NN with imperfect accuracy can provide better intuition behind the problem than overtrained one, whose predictions are impacted by overfitting. Used CNNs had accuracy between 89 and 96%.

For the results in Fig. 5.8 we apply transfer learning, which means that we use a NN that was trained with a training set that comes from a different domain than the test data. In our case the domains are different trajectories through the phase space indicated in Figure 5.5 with the arrows (1) and (2) where the phase transition appears for different values of V_1/J.

Physical Model: 1D half-filled spinless Fermi-Hubbard model

We apply the influence functions to the aforementioned CNN trained to recognize phases in the extended Hubbard model, namely a one-dimensional (1D) system consisting of spinless fermions at half filling. Hubbard models are of fundamental importance to the condensed-matter physics, with two-dimensional Fermi-Hubbard model believed to be a good toy model to describe the high-temperature superconductivity of cuprates [182]. The chosen 1D system, however, has the advantage of being within the power of efficient numerical simulations, and as a result, has a well-studied phase diagram [183, 184]. As such it is useful to present the power of influence functions (or any interpretability method) in phase classification problems. In this model, fermions hop between neighboring sites with amplitudes J as well as interact with nearest neighbors with strength V_1 and next-nearest neighbors with strength V_2:

$$\hat{H} = -J \sum_{\langle i,j \rangle} c_i^\dagger c_j + V_1 \sum_{\langle i,j \rangle} n_i n_j + V_2 \sum_{\langle\langle i,j \rangle\rangle} n_i n_j \,. \tag{5.3}$$

The competition between the system parameters J, V_1, and V_2 leads to four different phases: gapless Luttinger liquid (LL), two gapped charge-density-wave phases with density patterns 1010 (CDW-I) and 11001100 (CDW-II), and bond-order (BO) phase, as seen in Fig. 5.5. Two of the phases, namely the BO and CDW-II, co-exist in the limited range of parameters. Without the next-nearest-neighbor interaction, V_2, the system can follow only patterns of the gapless liquid Luttinger phase (LL) or the charge-density wave of the type I (CDW-I) with the degenerated density pattern 101010. The CDW-I order parameter describing this transition reads $O_{\text{CDW-I}} = \frac{1}{L} \sum_{\langle i,j \rangle} |n_i - n_j|$, where $\langle\rangle$ symbolizes nearest neighbors. We emphasize here that this order parameter captures the same behaviour as the $C_{\text{DW}}(i,j)$ in the BH model in 4.1, Equation 4.3. The next-nearest-neighbor interaction, V_2 competes with V_1, so for non-zero V_2 but still smaller than V_1 the transition between LL and CDW-I shifts towards bigger V_1.

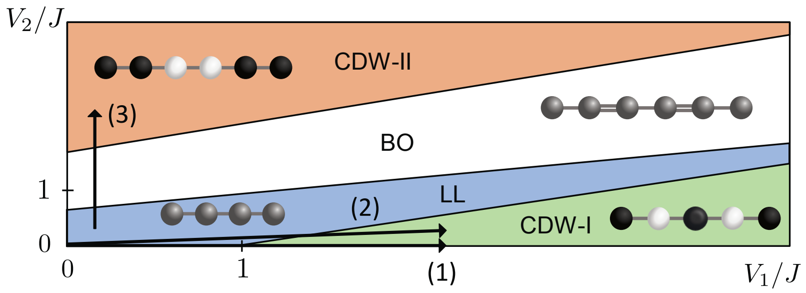


Figure 5.5: Schematic phase diagram of the extended one-dimensional half-filled spinless Fermi-Hubbard model with the schemes of the corresponding states: LL - Luttinger liquid, BO - bond order, CDW-I and II - charge density wave type I and II. The arrows indicate the transitions studied in this work.

For intermediate values of V_2 the bond-order (BO) phase emerges with the order parameter $O_{\text{BO}} = \frac{1}{L} \sum_i (-1)^i B_i$, where

$B_i = \left\langle c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right\rangle$. The BO phase is characterized by alternating strengths of the expectation value of the kinetic energy operator which leads to a spontaneous dimerization of the hopping between the sites. This is also illustrated in Figure 5.5 with the stronger couplings between alternating sites which illustrates this phase.

For large values for V_2 we obtain the charge-density wave of the type II (CDW-II) with the degenerated density pattern 11001100 for large V_2 values, with $O_{\text{CDW-II}} = \frac{1}{L}\sum_{\langle\langle i,j\rangle\rangle} |n_i - n_j|$, where $\langle\langle\rangle\rangle$ symbolizes next-nearest neighbors. The order parameters describing the transition to the CDW-I (-II) phase are the average difference between (next-)nearest-site densities.

To calculate the ground states and order parameters of the model, we use QuSpin package [185] to write the Hamiltonian for 12-site system in the Fock basis, resulting in 924 basis states. We assume periodic boundary conditions. The exact diagonalization is done with the SciPy package [186]. The ground states belonging to BO, CDW-I and II phases are degenerate. In order to lift the degeneracy of the ground state, we apply symmetry breaking fields that favor one of the patterns. For example the CDW-I phase can have the ground state density pattern 1010... and 0101.... To force the system to one of the two states, we add a field term $h_1 \sum_{i=1}^{N}(-1)^i n_i$ to the Hamiltonian which will reduce the energy of the density pattern 1010... and make it a non-degenrate ground state. The field is chosen $h_1 \ll 1$. This approach results in non-zero corresponding order parameters independently of the phase, therefore we define transition points as such parameters of the system that correspond to the order parameter being 10 times bigger than the corresponding symmetry breaking fields. As such, due to the guiding fields of values 10^{-7}, 10^{-5}, and 10^{-4} for 101010 and 11001100 density patterns and 1010 hopping pattern, respectively, the order parameters of values 10^{-6}, 10^{-4}, and 10^{-3} signal the transition to the CDW-I, CDW-II, and BO phase, respectively. It is interesting to note that the results presented in this work stay the same without the symmetry breaking fields, and also do not depend on the size of the system.

Within this work we train the CNN on three transition lines indicated with arrows (1)-(3) in Fig. 5.5. The first transition line leads from the LL to the CDW-I phase, and is calculated for constant $V_2 = 0$ and $V_1/J \in [0, 40]$. It is a source of training data for both Figs. 5.7 and 5.8, and test data for Fig. 5.7. It is symbolized in Fig. 5.5 with the arrow (1), and the values of corresponding order parameter $O_{\text{CDW-I}}$ are plotted in Fig. 5.6(a). The transition, defined as above, occurs for $V_1/J = 1$. The second transition line is calculated for $V_2 = 0.25\,V_1$ and $V_1/J \in [0, 80]$. Indicated with the arrow (2), it is the source of test data for Fig. 5.8. The corresponding order parameter CDW-I is plotted in Fig. 5.6(b), and the transition takes place for $V_1/J = 1.85$. The final transition line cuts three phases: LL, BO, and CDW-II. It is marked with the arrow (3) and provides both training and test data for Fig. 5.9. It is calculated for constant $V_1 = 1/J$ and $V_2 \in [0, 8V_1]$. Transition between LL and BO occurs for $V_2 = 0.51\,V_1$, and between BO and CDW-II for $V_2 = 1.7\,V_1$. It is important to notice that for the chosen range of parameters $V_2 = [1.7V_1, 8V_1]$, two phases co-exist what can be seen in Fig. 5.6(c). We feed

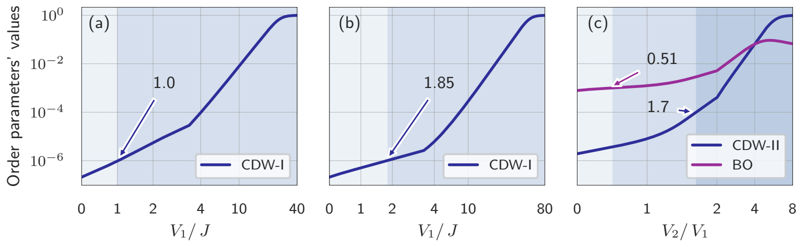


Figure 5.6: Corresponding order parameters' values for three transition lines studied within this work, indicated with arrows (1)-(3) in Fig. 5.5. (a)-(b) CDW-I order parameter for the transition line between the LL and the CDW-I phase for $V_2 = 0$ and $0.25V_1$, respectively. (c) CDW-II and BO order parameters for the transition line between LL, BO, and CDW-II for $V_1 = 1/J$. Note the logarithmic scale of y-axis, and the symmetric log scale of x-axis with threshold points chosen to be 3, 3, and 2, respectively. Cusps in the lines are artificial and result from the symmetric log scale of x-axis.

the CNN with ground states expressed in the Fock basis, labeled with their appropriate phases, calculated for a 12-site system with QuSpin and SciPy packages [185, 186]. To lift the degeneracy of the ground state, we use guiding fields favoring one symmetry. As shown above, we define the phase transition position where the order parameter's value is ten times larger than the corresponding guiding field. The hopping amplitude, J is set to 1 throughout the paper.

Data and code availability

The code needed to reproduce all results, along with the data production, is available for further development and use on GitHub [180].

5.2 Results

In this section we present the results for the influence function applied on the different transitions within the extended Fermi-Hubbard model.

Transition between LL and CDW-I.

We train a CNN to classify ground states into two phases: LL and CDW-I based on the transition line marked with the arrow (1) in Fig. 5.5 for $V_2 = 0$. We plot the influence functions of all training examples for a chosen test point (marked with orange line) in Fig. 5.7. The order parameter describing the transition here is the average difference between nearest-site densities, which is zero in the LL phase and non-zero (growing to one) in the CDW-I phase. The panels (a)-(b) present how influential training points are for test points from the LL phase. The test state (a) is the ground state located deeply in the LL phase, while (b) is closer to the transition.

If the CNN learns an order parameter, all training points, i.e., ground states from the LL phase exhibiting a zero order parameter, should be similarly positively influential, and that is exactly what we observe. They form an almost flat line in panels (a), (b), and (d). In panel (c), however, for the test point close to the transition, their influence changes linearly. This divergence from expected behavior is because, in our exact diagonalization calculations, the order parameter in the LL phase is not exactly constant and equal to zero. Instead, it is growing very slowly, that is why finally the most helpful points are the ones near the transition - they are also the most unique from the training points labeled as LL, and the information they provide is the most valuable. The nonzero order parameter is caused by three phenomena: the finite-size effect, use of the guiding fields, and the numerical arbitrariness of choosing the transition point. In the perfect scenario (observed, for example, for training on states obtained from mean-field calculations), the five most influential

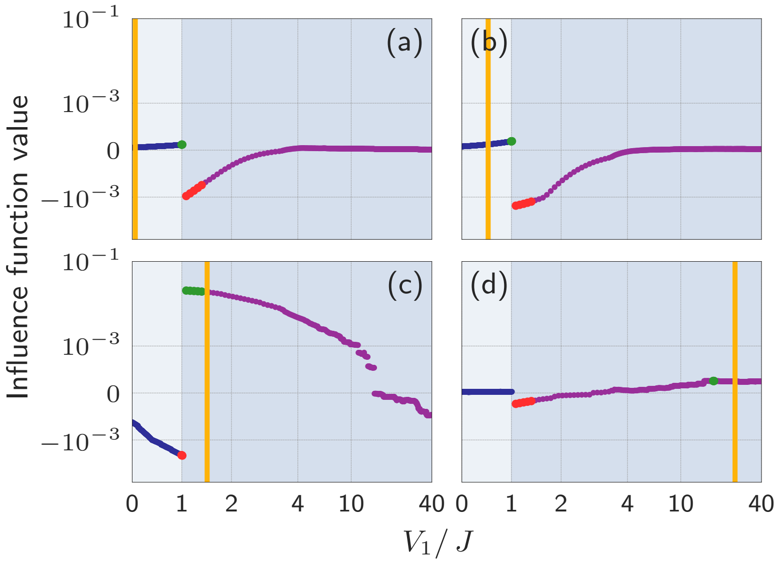


Figure 5.7: Influence functions of all training examples i.e., ground states calculated for the transition line between LL and CDW-I for $V_2 = 0$ for chosen test points marked with an orange line. Blue dots are influence function values for training examples from the LL phase; purple ones are from the CDW-I phase. Larger green (red) dots are five the most influential helpful (harmful) training examples. The light grey background indicates V_1/J values of the LL phase, dark grey - the CDW-I phase. A symmetric log scale is used both in x and y axis with 3 and $|10^{-3}|$ chosen as threshold points, respectively.

points should be randomly distributed over the whole LL phase. The most harmful training points are, in both cases, the ones closest to the transition, but on the CDW-I side of it. These states are the most similar (with the smallest order parameter value), but already labeled differently. On the side of the CDW-I phase, the influence pattern is significantly different. The curvature of influential points corresponds to the growth of the order parameter, and the most influential helpful points are the ones closest to the test point in the order parameter space, slightly shifted towards the transition point, as they provide more information. Panel (c) shows the influence functions of training points for the test states on the CDW-I side, close to the transition. The most harmful examples are, as in the previous test points, the ones closest to the transition, but on the other side of it. Panel (d), however, presents a distinct behavior of the most harmful examples being the closest to the transition, but on the same side. All the training points are similarly influential with small values of influential functions resulting in the almost flat line. It is a signature of the CNN's high certainty regarding the prediction made in panel (d) manifesting with a small test loss function $\mathcal{L}(z_{\text{test}}, \hat{\theta})$. Also, the analyzed test point is deeply in the CDW-I phase with all neighboring states being almost identical with the order parameter close to 1. The most harmful examples are the ones that are labeled as the CDW-I phase, but very different, so the ones closest to the transition. Important to keep in mind is that the training data provided to the machine are fully shuffled, so the ordering, which is visible in the figures, results solely from their inner analysis by the ML algorithm.

Transfer learning

With a similar approach, we validate the transfer learning to another transition line. We take the trained CNN from Fig. 5.7 and in Fig. 5.8 we apply it to test states coming from the transition line for $V_2 = 0.25\,V_1$, where the phase transition position is shifted to higher values of V_1/J. Therefore the training and test states come from different transition lines, $V_2 = 0$ and $0.25\,V_1$, marked in Fig. 5.5 with the arrows (1) and (2), respectively.

Panels (a) and (b) of Fig. 5.8 show the influence function values of training data set for test states from the LL phase, while (c) and (d) - from the CDW-I phase. Due to the shifted transition point for the test line, compared to the training line, we see the same shift in the behavior of the most influential points on the CDW-I side. This shift corresponds to the fact that no longer the same value of V_1/J yields the same value of the order parameter, and that the ML algorithm still as the most influential points regards the states with the most similar order parameter.

Inferring the existence of the third phase

This time we analyze the transition line crossing three phases, LL, BO, and CDW-II, which is indicated by the arrow (3) in Fig. 5.5. Two order parameters are needed to describe this transition. One is the average difference of the next-nearest neighbor density, which equals zero in the LL and BO phases, and grows to 1 in the CDW-II phase. The other is the staggering of effective

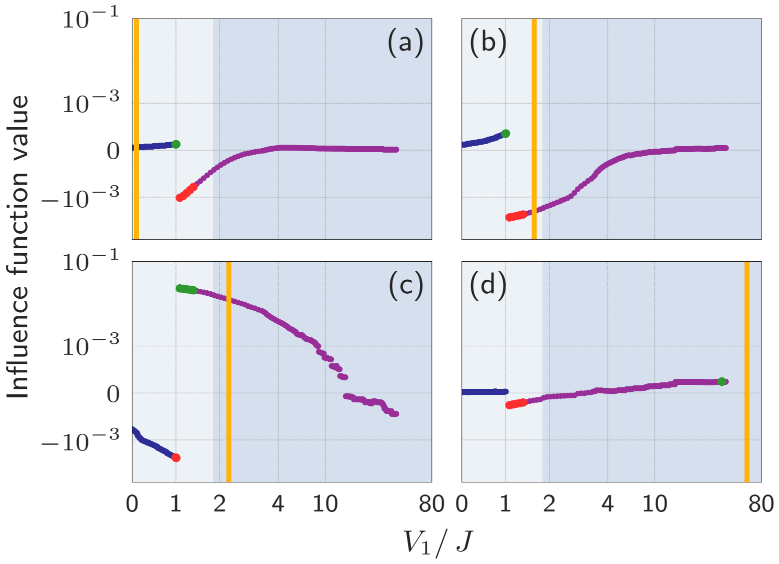


Figure 5.8: Influence functions of all training examples i.e., ground states calculated for the transition line between LL and CDW-I for $V_2 = 0$ for chosen test states from transition line for $V_2 = 0.25\,V_1$ marked with an orange line. Blue dots are influence function values for training examples from the LL phase for $V_2 = 0$; purple ones are from the CDW-I phase. Larger green (red) dots are five the most influential helpful (harmful) training examples. The light grey background indicates V_1/J values of the LL phase for $V_2 = 0.25\,V_1$ line, dark grey - the CDW-I phase. A symmetric log scale is used both in x and y axis with 3 and $|10^{-3}|$ chosen as threshold points, respectively.

nearest-neighbor hoppings, being 0 in the LL phase, non-zero in the BO phase, and slowly decaying to 0 in the CDW-II phase. In the studied range of parameters, two phases (BO and CDW-II) co-exist. It is crucial to note that in this section, we train on the mentioned transition line crossing three phases, but we label ground states only as belonging to one out of two phases.

In the first set-up, with results presented in the panels (a)-(b) of Fig. 5.9, we label ground states as belonging to the LL (blue dots, label 0) or belonging to the BO and CDW-II phases (purple dots, label 1). Independently on the test point location, within purple training points belonging to BO and CDW-II two similarity regions, understood as two groups of points with similar influence within the group, can be distinguished. The ML algorithm apparently learns two different patterns (order parameters)

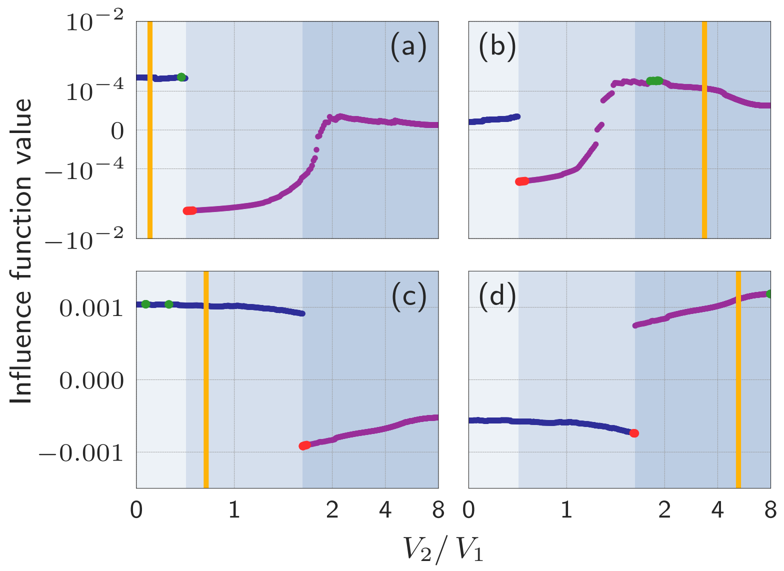


Figure 5.9: Influence functions of all training examples i.e., ground states calculated for the transition line between LL, BO, and CDW-II for $V_1/J = 1$ labeled as (a)-(b) LL - not LL and (c)-(d) CDWII - not CDWII for chosen test points marked with an orange line. Blue dots are influence function values for training examples from (a)-(b) the LL phase, (c)-(d) the LL and BO phases, while purple ones (a)-(b) from the BO and CDW-II phases, (c)-(d) from the CDW-II phase. Larger green (red) dots are five the most influential helpful (harmful) training examples. The light background indicates V_1/J values of the LL, light grey - BO, and dark grey - CDW-II phase. In all subplots, a symmetric log scale is used in x axis with 2 chosen as a threshold point. A symmetric log scale with $|10^{-4}|$ as a threshold is used in y axis in panels (a)-(b), while in panels (c)-(d) the scale is linear.

to classify the data correctly, and as such, it notices the existence of the third phase within the incorrectly labeled data. This would be impossible to notice without the use of interpretability methods, which in this sense pave the way towards the detection of unknown phases. The second set-up consists of labeling the same data as belonging either to the LL and BO phases (blue dots, label 0) or to the CDW-II phase (purple dots, label 1). The influence functions values, resulting from this classification, are in the panels (c)-(d) of Fig. 5.9. The pattern they form is starkly different. First of all, no longer two similarity regions within training points from the LL and BO phases are distinguished. It is because this transition can be fully described with one order parameter, which is zero in the LL and BO phases, and non-zero in the CDW-II phase. The behavior is then more similar to the one seen in Fig. 5.7 with the transition between LL and CDW-I. It is not identical, though, as in the phase LL+BO the most helpful training points are always distributed randomly, but deep in the LL phase, avoiding the BO phase. The most helpful points on the CDW-II side are also those deep in the CDW-II phase in contrast to Fig. 5.7, where they mostly follow the test point. The difference comes mostly from the fact that the deeper in the CDW-II phase, the smaller the BO order parameter, which is making CDW-II predictions less difficult. We claim that the observed pattern is the sign of not learning correctly the order parameter and potentially overfitting.

Finally, we trained a CNN on the same data, but with three labels correctly corresponding to all three phases. The influence patterns seen in Fig. 5.7 and panels (c)-(d) of Fig. 5.9 are repeated, indicating that CNN correctly learns both appropriate order parameters.

5.3 Conclusion

We used an interpretability method, the influence functions, on a CNN trained to classify ground states of the extended 1D half-filled spinless Fermi-Hubbard model. We provided for the first time a strong indication that the ML algorithm indeed learned a relevant order parameter. First, in the LL - CDW-I transition the order parameter is zero in LL and continuously increasing in CDW-I. Therefore, the states in LL should all be all equally influential. In the CDW-I phase, on the other side, the points that are close in the parameter space should be most similar to each other and with increasing distance in the parameter space the influence should decay. Exactly this behaviour can be obtained with the influence function. At the same time we also have to acknowledge significant finite size effects that show in the order parameter and in the influence function. E.g. the order parameter is not exactly zero in the LL phase.

We as well show that in a transfer learning setup, where the LL-CDW-I phase transition for the test set is slightly shifted, the influence function still identifies influential training points in accordance of the new shifted transition.

Second, we train the CNN on states from three different phases, but only provide two different labels. From this experiment we can obtain, that the CNN learns by itself to distinguish BO from CDW-II if these two phases are labeled equally and compared against the LL phase. This makes sense, because there are two different order parameters for BO and CDW-II that are both zero in LL. Therefore, when we tell the CNN that it has to distinguish BO and CDW-II against LL, we can see in the influence function, that BO and CDW-II are distinguished differently from LL. Which is the same if we would apply an order parameter. If, on the other side, we train the CNN on LL and BO versus CDW-II, the CNN only has the possibility to distinguish the two classes via the CDW-II order parameter, which compares the density the second nearest neighbour sites. This order parameter is zero in BO and LL and therefore via this order parameter these two phases are not distinguishable. This shows as well in the influence function.

Chapter 6

Loss Landscapes of VQCs

In Chapter 5 we have seen how the eigenvalues of the Hessian of a loss function can help us to calculate the influence function to get more insight into the black-box of NNs. In recent years Noisy intermediate-scale quantum (NISQ) devices [27] have drawn a lot of attention. Variational quantum circuits (VQCs) are one of the most promising applications of NISQ devices and like classical NNs they are used as black-boxes. In this Chapter, we introduce a way to compute the Hessian of the loss function of VQCs and show how to characterize the loss landscape with it. The eigenvalues of the Hessian give information on the local curvature and we discuss how this information can be interpreted and compared to classical NNs. We benchmark our results on several examples, starting with a simple analytic toy model to provide some intuition about the behavior of the Hessian, then going to bigger circuits, and also train VQCs on data. Finally, we show how the Hessian can be used to escape flat regions of the loss landscape. Even though VQCs have been benchmarked on many different problems in the past years, there are still many open questions related to the convergence of the loss function and to the trainability of these circuits in situations of vanishing gradients. Furthermore, it is not clear how "good" the minima are in terms of generalization and stability against perturbations of the data and there is, therefore, a need for tools to quantitatively study the convergence of the VQCs.

VQCs are constructed with parametrized gates to minimize a given observable (or measurement) and can be trained with gradient-based methods [187]. There are two main avenues for the use of VQCs. The first one is the use of VQCs as a state ansatz with the purpose of finding the variational parameters that minimize a given observable, such as the energy of a physical system for the Variational Quantum Eigensolvers (VQE) [188, 189] or Quantum approximate optimization algorithms (QAOA) [36]. Such circuits allow one to study for example chemistry problems [35] as well as classical optimization problems, where optimization tasks can be mapped to spin problems [36]. The second main application for VQCs encompasses data driven quantum machine learning (QML) tasks and their purpose is to map classical input data onto a measurement that serves as a label [190, 191, 192]. The emphasis in this application is to predict the right label for different classical inputs and the quantum states generated by VQCs are high dimensional representations of the classical data. These architectures of VQCs have shown to be universal approximators [191] and are therefore well suited to do ML. We will refer to this second application as Quantum Neural Network (QNN). Both of these applications have a classical NN analogue. NNs have been used as variational ansatz to represent quantum states [193, 194] and, like classical NNs, QNNs can be trained on data and do classification or even work as generative models [195].

Recent works in VQCs were mainly devoted to the exploration of new applications. Nevertheless, the benchmark of the performance and advantage of such circuits compared to classical NNs has not yet received the deserved attention. In this chapter, we focus on the loss landscape of VQCs and characterize it with the help of the Hessian of the loss function. The loss landscapes of classical NNs trained on data have been extensively studied and we here compare the loss landscape of data driven classical NNs and QNNs. For data driven classical NN tasks, the study of the loss is almost as old as NNs themselves: it started with the investigation of spin glass models for a better understanding of Hopfield networks [196] and is still a very active research area. For example, numerical studies of over-parametrized classical NNs suggest that high error local minima traps do not appear [197, 198] in these high dimensional optimization landscapes. Empirical evidence even suggests that low loss basins in NNs are connected and there are no barriers separating them from each other [199]. Furthermore, different training algorithms find comparable solutions with similar training accuracies [200]. These findings are all in contradiction with the original idea of a spin-glass-like loss landscape, where many low energy solutions exist and one can get trapped in spurious minima. Supervised NN setups show good performance in generalizing tasks: once trained on data, they are able to predict labels of unknown test data. It is, until today, still the subject of debates what kind of topology the loss minimum of a well trained NNs should have. In Ref. [201], the authors showed that even though the training is independent of the gradient descent method, the generalization of the NNs depends on many hyper parameters, such as the batch size during the training: Smaller batch sizes tend to generalize better leading to wider minima basins of the loss function with many zeros eigenvalues of the Hessian $\lambda_i = 0$ and almost none of them negative $\lambda_i < 0$. These findings support the idea that wider minima generalize better, whereas this question is not yet finally answered and is still subject to today's research in classical ML. The nature of the non-convexity of classical NNs is still in the focus of ongoing research [202].

A new phenomenon and a major short coming of VQCs compared to classical NNs is the occurrence of Barren Plateaus [203, 204, 205], which are characterized by flat plateaus of the loss function and exponentially vanishing gradients. In particular, the authors of Ref. [204] showed that the appearance of the Barren plateaus depends on the choice of the cost function and that for local cost functions, gradient decays only with a power law. The authors also claimed that the loss landscape of certain problems resembles a narrow gorge, which somehow contrasts with the idea of wide basins for good generalization in classical NNs. In Ref. [205], the authors showed that one can avoid to initialize the variational parameters of the circuit in a Barren Plateau. In so-called hybrid-quantum-classical training settings, standard gradient methods like Stochastic gradient descent (SGD), the ADAM optimizer [206] or also the quantum natural gradient (QNG) have been used to train VQCs. QNG showed good convergence [187] and also helped to avoid local minima [207], which is in contrast to the aforementioned independence of the training method for classical NNs. In Ref. [191], the authors used a Hessian based optimization method and obtain faster convergence for certain data driven problems. Later in this chapter, we will provide a possible explanation for this phenomenon.

Generally the loss landscape of VQCs is not yet well understood and, as for classical NNs, a better understanding might lead to the improvement of optimization algorithms and show the limitations of certain circuit designs. In this chapter we show how the loss landscape of different VQCs can be studied via the eigenvalues of the Hessian.

This Chapter is structured as follows: In Sections 6.1, we start with a brief introduction of the Hessian of NN losses, the loss landscape of classical NNs and the distribution of the eigenvalues of the Hessian. In Section 6.2, we discuss the loss landscapes of VQCs. In Section 6.3, we study an analytical example of a VQC and compute the Hessian and its eigenvalues. We show how to calculate the Hessian on an actual quantum hardware in Section 6.4, apply it to a general example in Section 6.5 and study it numerically. In Section 6.6, we study a VQC trained on classical data acting as a classifier. Finally in Section 6.7, we show how the Hessian can help to escape from very flat barren-plateau-like regions in the landscape.

6.1 Charaterization of the Loss Landscape with the Hessian

Hessian and Curvature

The Hessian of a function is a well-suited mathematical object to study its the local curvature. Given a function $f(\boldsymbol{\theta}) : \mathbb{R}^N \to \mathbb{R}$, the Hessian is defined as the square matrix of the second derivatives of this function $H_{ij} = \partial_{\theta_i}\partial_{\theta_j} f(\boldsymbol{\theta})$, with $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_N)$. For functions satisfying the Schwartz theorem, the Hessian is a symmetric matrix, so its eigenvalues λ_i are real and can be ordered $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. We denote by $\mathbf{v_i}$ the eigenvector associated with the eigenvalue λ_i and refer to the pair $(\lambda_i, \mathbf{v_i})$ as the i-th eigenpair.

The Hessian $H|_{\boldsymbol{\theta}}$ evaluated at a certain point $\boldsymbol{\theta}$ gives a second order approximation of the function $f(\boldsymbol{\theta})$ within Taylor's expansion and its spectrum allows one to extract information about the local curvature of $f(\boldsymbol{\theta})$: A positive eigenvalue $\lambda_i > 0$ of an eigenpair indicates a locally positive curvature in the direction of $\boldsymbol{v}_i$ and, therefore, an increase of $f(\boldsymbol{x})$ if one moves along the direction $\boldsymbol{\theta} + \epsilon \mathbf{v}_i$, for a small perturbation ϵ. Analogously, negative eigenvalues indicate a locally negative curvature and $\lambda_i = 0$ indicate flat directions of the function $f(\boldsymbol{\theta})$ and, consequently, zero curvature. Accordingly, if the function is at an extremum $\nabla f(\boldsymbol{\theta}) = 0$ and all eigenvalues are positive (resp. negative), the point is a local minimum (maximum). If some eigenvalues are positive and some are negative the extremum is a saddle point.

Loss landscape of neural networks: a brief review

The training of a NN in a supervised ML setting consists in minimizing the empirical risk $\mathcal{L} = \sum_i^N l(f(\boldsymbol{x}_i, \boldsymbol{\theta}), y_i)$ over a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_i^N$, where x_i is a training point with its corresponding label y_i. $f(\boldsymbol{x}_i, \boldsymbol{\theta})$ is the NN prediction for the training point $\boldsymbol{x}_i$ parameterized by the weights $\boldsymbol{\theta}$. The loss function $l(\cdot)$ measures the difference between the NN prediction and the label. Commonly used loss functions are the mean square error or the cross entropy [208]. The Hessian is defined as $H_{ij} = \partial_{\theta_i\theta_j}\mathcal{L}$ and has a wide range of applications in ML: It can be used to adapt gradient update to the current loss landscape in the so-called "Newton" method [209], for pruning [210, 211] or for interpretability purposes with the influence function [212]. Furthermore, it can also be used to study the local curvature of the loss for a better understanding of the loss landscape and the convergence of NNs. In an extensive empirical study, the authors of Ref. [200] charaterized the behaviour of the Hessian for over-parameterized NNs, where the number of weights $\boldsymbol{\theta}$ of the NN is much bigger than the number of training points N. They showed that, after training, the eigenvalues of the Hessian are distributed such that most eigenvalues are in a bulk close to zero and that there are some outliers away from zero. For a randomly initialized NNs, the outliers are both positive and negative and they are symmetrically distributed around zero. With the training progress, the amount of positive eigenvalues increases and it converges to a negligible amount of negative eigenvalues. For a trained NNs most eigenvalues are zero and a few are positive, which indicates a flat minimum of the loss landscape. These findings suggest that classical notions of basins of attraction may be misleading and the minima of over parameterized NNs are extremely flat pools with just a few steep directions. The authors also showed that the number of eigenvalues that are significantly larger than zero can be even smaller than the number of training points N given that the training data instances of each class do not deviate significantly. They can be of the order of k, which is the number of classes that have to be learned by the NN.

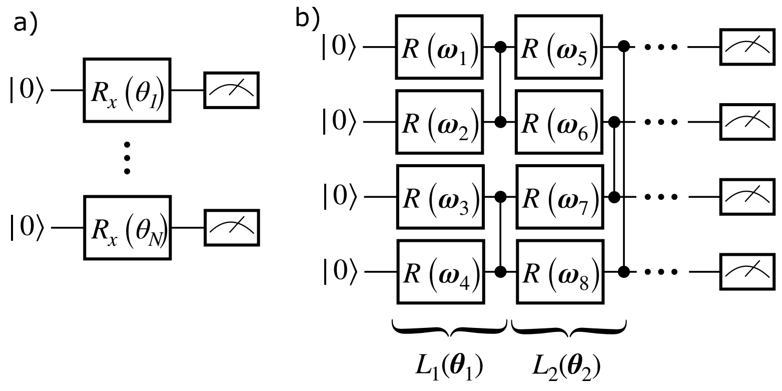

a)
$|0\rangle$
$R_x(\theta_1)$
$|0\rangle$
$R_x(\theta_N)$
b)
$|0\rangle$
$R(\boldsymbol{\omega}_1)$
$R(\boldsymbol{\omega}_2)$
$R(\boldsymbol{\omega}_3)$
$R(\boldsymbol{\omega}_4)$
$R(\boldsymbol{\omega}_5)$
$R(\boldsymbol{\omega}_6)$
$R(\boldsymbol{\omega}_7)$
$R(\boldsymbol{\omega}_8)$
$L_1(\boldsymbol{\theta}_1)$
$L_2(\boldsymbol{\theta}_2)$

Figure 6.1: Variational Circuits. a) Toy model circuit with 1 qubit rotations around the x axis for N qubits without entangling gates. b) A general 4 qubit circuit with 1 qubit rotations $R(\boldsymbol{\omega}_i) = R(\phi_1, \phi_2, \phi_3)$ followed by CZ entangling gates. We denote by L_i is a combination of rotations and entangling gates.

Computational methods

For most of the implementations we use the pennylane package from Ref. [213]. We also wrote our own code in pytorch to speed up experiments with a big number of parameters. For the implementation we used the complex pytorch library from Ref [214]. All code can be found on GitHub [215].

6.2 Loss Landscape of VQCs

For the study of loss landscapes of VQCs the literature, so far, is rather sparse and there are still many open questions. For pure quantum optimization tasks, such as QAOA, VQE or also state preparation, in general, Barren plateaus seem to be a major shortcoming and $\nabla f(\boldsymbol{\theta}) = 0$ might not indicate an extremum in the loss landscape but an absolutely flat region, where all eigenvalues of the Hessian are zero. In Ref. [204], the authors defined a Barren Plateau as a point in the parameter space, where the mean gradient $\langle \nabla f(\boldsymbol{\theta}) \rangle = 0$ and the variance $\mathrm{Var}\left[\nabla f(\boldsymbol{\theta})\right]$ exponentially vanish with the number of qubits N for global cost functions and any circuit depth d. They also showed that Barren plateaus can be avoided for local cost functions if the circuit depth is $d \in \mathcal{O}(\log(N))$. For data driven tasks, where VQCs are trained on classical data, which we will refer to as a QNN, the occurrence of Barren Plateaus has been observed in some specific architecutres [216]. In section 6.6, we investigate a QNN which seems to not suffer from vanishing gradients. The latter might come from the fact that such architecture escapes the ones described in Ref. [216]. The loss landscape for VQCs has so far been studied in the context of the Quantum Natural Gradient (QNG) [187, 207] where it has been shown that including the local curvature of the quantum state can improve training convergence and also help to avoid local minima. Furthermore, in VQC settings, where the aim is to find a target state $|\psi_T\rangle$, the question of how expressive a certain circuit ansatz is and whether the ansatz contains a solution to the problem becomes relevant [217]. Unfortunately, if one wants to find a target state with gradient based optimization methods, it will not be enough to know that the ansatz contains a solution. It might be impossible to reach it from a certain initialization of the parameterized circuit, because one gets stuck in local minima. This problem has been studied for classical NNs and empirical evidence suggests that for overparametrized NNs the local minima basins are connected [218] and therefore overparametrization is as crucial as the bare existence of the solution itself. It is still unclear if these notions of overparameterization also apply to VQCs. Hence, there is a need for a better understanding of the loss landscapes of general VQCs and we believe that the use of the Hessian is one possible tool to achieve this.

6.3 Loss Landscape of VQCs: An Analytical Example

In this section we analytically characterize the curvature of the loss landscape with the Hessian. We start with a simple circuit that can be solved analytically to get familiar with the concept of the Hessian of loss functions of VQCs. We choose a toy model circuit introduced in Ref. [204] that presents Barren Plateaus. The toy model is a N qubit VQC $V(\boldsymbol{\theta}) = \otimes_i^N R_x(\theta_i) =$

$\otimes_i^N \exp(-i\theta_i/2\sigma_x)$, shown in Fig. 6.1(a), with randomly initialized parameters $\boldsymbol{\theta}$ that generates the state

$$V(\boldsymbol{\theta})\,|\mathbf{0}\rangle = \sum_{k=0}^{n}(i)^k P\left(\prod_i^{n-k}\cos(\frac{\theta_i}{2})\prod_j^{k}\sin(\frac{\theta_j}{2})\,|0\rangle^{n-k}\,|1\rangle^{k}\right), \tag{6.1}$$

where $|\mathbf{0}\rangle \equiv \otimes_i^N |0\rangle_i$ and $P(\cdot)$ stands for the sum of all possible permutations of the argument. For example

$$\begin{aligned} &P\left(\cos(\theta_1)\sin(\theta_2)\,|01\rangle\right)\\ &= \cos(\theta_1)\sin(\theta_2)\,|01\rangle + \cos(\theta_2)\sin(\theta_1)\,|10\rangle\,. \end{aligned} \tag{6.2}$$

The aim of the variational algorithm is to rotate the initial state $V(\boldsymbol{\theta})\,|\mathbf{0}\rangle$ into a given target state $|\psi_T\rangle$ and to maximize the fidelity $\mathcal{F}$ with respect to $|\psi_T\rangle$. For the particular target state $|\psi_T\rangle = |\mathbf{0}\rangle$, all the $\sin(\cdot)$ terms in the variational state of Eq. (6.1) cancel and the fidelity reads [204]

$$\mathcal{F} = |\,\langle\psi_T|\,V(\boldsymbol{\theta})\,|\mathbf{0}\rangle\,|^2 = \prod_i^n \cos^2\left(\frac{\theta_i}{2}\right). \tag{6.3}$$

Therefore, we can translate this optimization problem into the minimization of the loss function $l = 1 - \mathcal{F}$, shown in Fig. 6.2, as a function of θ_1 and θ_2 and for $\theta_{i>2} = 0$.

We now discuss how the Hessian can help to understand the loss landscape.

First we initialize the parameters $\boldsymbol{\theta}$ randomly and we want to find the parameters to generate the target state. Figure 6.2 depicts the eigenvalues of the Hessian for different values of θ_1 and θ_2. The points a) - d) in the optimization landscape show a possible trajectory of an optimizer. Point a) shows an initialization in a Barren Plateau, where all the eigenvalues of the Hessian are 0. Points b) and c) are saddle points in this high dimensional optimization space where some of the eigenvalues are negative, some are positive and the bulk of them is zero. Point d) shows a well converged loss with no negative eigenvalues of the Hessian and zero gradient ensuring that it is indeed a local minimum. Since the loss $l = 0$, we even know that it is a global minimum. We also observe many degenerate zero eigenvalues which implies that the minimum is not isolated. As it was the case for classical NNs, this is a consequence of the over parametrization of the

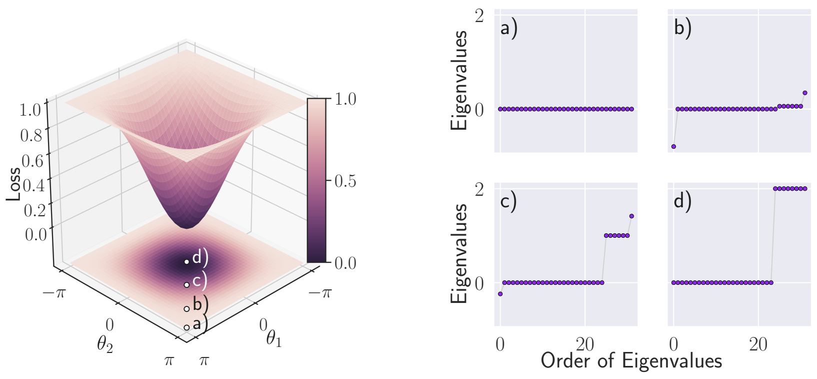


Figure 6.2: Loss landscape of the toy model with a global loss function. The target state is chosen to be $|\psi_T\rangle = |\mathbf{0}\rangle$. We label in the contour plot the points for which we compute the Hessians and its eigenvalues. For fixed parameters $\theta_i = 0$ with $i > 2$ the loss function is not dependent on the number of qubits. Panel a) shows the vanishing eigenvalues which indicate the Barren Plateau described in Ref. [204].

VQC: many different linear combinations of angles lead to the same loss function.

For two free parameters θ_1 and θ_2 and the rest fixed, we find that the amplitudes of the eigenvalues of the Hessian do not depend on the system size. This might seem to contradict the phenomenon of narrow gorge described in Ref. [204], where the authors describe that the valley in Fig. 6.2 becomes narrower with an increasing number of qubits N. But in their work, they change half of the parameters collectively along one axis and the other half along the other axis. Therefore, it is still true that the VQC is more likely to be initialized in a flat region even with our description, because the parameter space becomes bigger and the product of sinusoidal functions of Eq. (6.1) becomes smaller for a larger number of randomly drawn parameters.

Figure 6.3 shows the loss landscape for a local loss function $l = 1-\sum_i |\langle\Psi|0\rangle_i|^2$, where $|\Psi\rangle = V(\boldsymbol{\theta})|\mathbf{0}\rangle$ is the variational state and $|0\rangle_i$ is the qubit state of qubit i. In contrast to the global loss of Eq. (6.3), we measure each qubit separately. For the local loss, we find that there is no point in the loss landscape with vanishing eigenvalues of the Hessian .

Note that this behaviour also applies to circuit ansatzes that do not contain the exact target state and converge at high losses. If we choose the equal superposition as a target state $|\psi_T\rangle =$

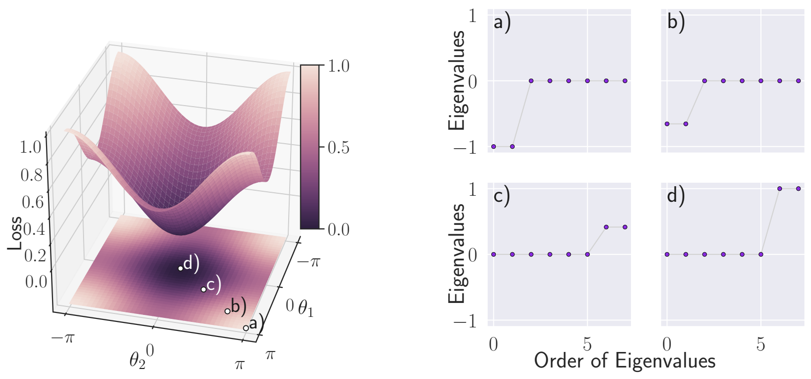


Figure 6.3: Loss landscape of the toy model with a local loss function. We label in the contour plot the points for which we calculate the Hessian and its eigenvalues. The loss landscape is shown here for only 2 qubits, because for the local loss if we have more than 2 qubits and fix their rotational parameters $\boldsymbol{\theta}$ for the 3D loss, we will not obtain the full range variation of the loss from 0 to 1. The landscape looks qualitatively the same for more qubits.

$\sum_{\{\boldsymbol{\sigma}\}} |\boldsymbol{\sigma}\rangle$ the circuit $V(\boldsymbol{\theta})$ that only consists of Pauli X rotations will not be able to rotate the initial state $|0\rangle$ such that it matches the target state with fidelity 1. However, with an eigenvalue pattern like in Fig. 6.4, we know that point d) is a local minimum, given that all the eigenvalues are either positive or zero. It becomes also clear that zero eigenvalues of the Hessian correspond to directions where changes in parameters do not affect the loss landscape. This phenomenon is illustrated in Figure 6.4 the direction $(\theta_1, -\theta_2)$ moves along a valley with constant loss. Therefore, the eigenvectors of the Hessian reveal additional information which help to find directions in the loss landscape of maximum or minimum stability. The latter might be used for pruning the network [210].

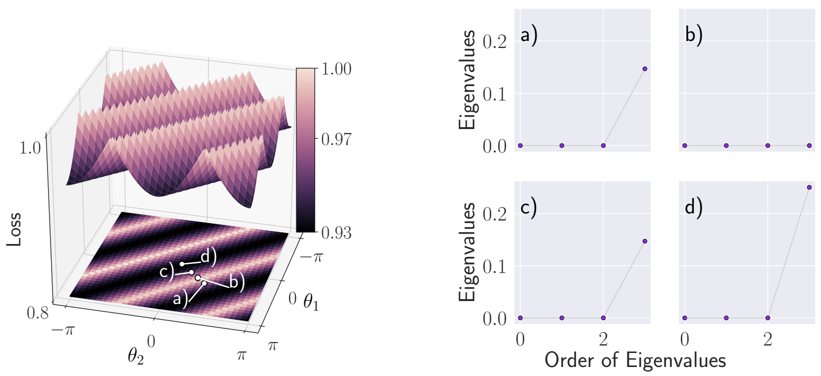


Figure 6.4: Loss landscape of the toy model with $|\psi_T\rangle = \sum_{\{\boldsymbol{\sigma}\}} |\boldsymbol{\sigma}\rangle$. In this case, the circuit is under-parametrized and cannot reach minimum loss. Nevertheless, we can read from the Hessian's eigenvalues that we reached a stable minimum. Furthermore, one of the eigenvectors of the Hessian with eigenvalue $\lambda = 0$ is $\boldsymbol{v} = (\theta_1, -\theta_2)$. Along this direction, the loss is constant. The latter can be verified in the contour plot.

6.4 Computation of the Hessian of a Quantum Circuit

Like in classical ML, we can treat the VQC as a black-box function that takes classical data $\boldsymbol{x}$, depends on some learnable parameters $\boldsymbol{\theta}$ and returns some classical value $f(\boldsymbol{\theta}, \boldsymbol{x})$ from a measurement or some expectation value. The gradient of any quantum circuit can be calculated by estimating the central finite difference of the circuit output

$$\partial_{\theta_i} f(\boldsymbol{\theta}, \boldsymbol{x}) = \lim_{\varepsilon_i \to 0} \frac{f(\boldsymbol{\theta}_{\neg i}, \theta_i + \varepsilon_i, \boldsymbol{x}) - f(\boldsymbol{\theta}_{\neg i}, \theta_i - \varepsilon_i, \boldsymbol{x})}{2\varepsilon_i}, \quad (6.4)$$

where $\boldsymbol{\theta}_{\neg i}$ are all the parameters except θ_i. To approximate the limit of $\varepsilon_i \to 0$ on real hardware, one should choose $\varepsilon \ll 1$. This kind of gradient estimation has been shown to not perform well and depends strongly on the stochasticity of the measurement outcomes, the number of measurements and the choice of ε [219]. Because of the factor $\frac{1}{\varepsilon}$, the measurement noise can be amplified and the estimation of the gradient requires more measurement shots for smaller ε. In Refs. [220, 221], the authors showed how the gradient can be calculated analytically to avoid the dependency on the hyperparameter ε. The authors in [221] focused

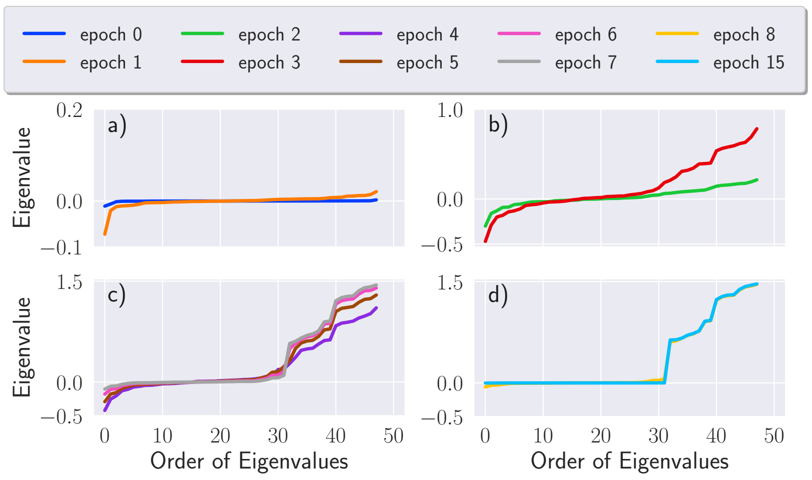


Figure 6.5: Eigenvalue evolution during the training. We separate the curves for different epochs to increase the visibility as the difference between the smallest and biggest eigenvalues is varying during the training.

on quantum circuits that can be implemented on NISQ devices, where the parametrized gates are any qubit rotations generated by Pauli operators. Taking advantage of these gates, the analytic gradient reads

$$\partial_{\theta_i} f(\boldsymbol{\theta}, \boldsymbol{x}) = \frac{1}{2} \langle f(\boldsymbol{\theta}_{\neg i}, \theta_i + s, \boldsymbol{x}) \rangle - \langle f(\boldsymbol{\theta}_{\neg i}, \theta_i - s, \boldsymbol{x}) \rangle, \qquad (6.5)$$

where we set $s = \pi/2$. This is called the parameter shift rule and $s = \pi/2$ for all qubit gates that are generated by matrices with eigenvalues equal to $\pm 1/2$, for example Pauli spin-$1/2$ matrices. To obtain the gradient of a loss function $l(f(\boldsymbol{\theta}, \boldsymbol{x}))$, we apply the chain rule $\partial_{\theta_i} l(f(\boldsymbol{\theta}, \boldsymbol{x})) = \partial_{\theta_i} f(\boldsymbol{\theta}, \boldsymbol{x}) l'(f(\boldsymbol{\theta}, \boldsymbol{x}))$, where l' is the first derivative of the loss function with respect to the argument $f(\boldsymbol{\theta}, \boldsymbol{x})$.

As proposed in [222], the Hessian of a quantum circuit can be computed by applying the parameter shift rule twice

$$\begin{aligned}\partial_{\theta_j}\partial_{\theta_i} f(\boldsymbol{\theta},\boldsymbol{x}) =& \frac{1}{4}\ \big[\ \langle f(\boldsymbol{\theta}_{\neg i,j},\theta_i+s,\theta_j+s,\boldsymbol{x})\rangle \\ &+ \langle f(\boldsymbol{\theta}_{\neg i,j},\theta_i-s,\theta_j-s,\boldsymbol{x})\rangle \\ &- \langle f(\boldsymbol{\theta}_{\neg i,j},\theta_i-s,\theta_j+s,\boldsymbol{x})\rangle \\ &- \langle f(\boldsymbol{\theta}_{\neg i,j},\theta_i+s,\theta_j-s,\boldsymbol{x})\rangle\big].\end{aligned} \tag{6.6}$$

With this tool we are able to study the curvature of a loss function $l(f(\boldsymbol{\theta},\boldsymbol{x}),y)$ via the second derivative of the loss. We emphasize here that one has to apply the chain rule twice to obtain the correct Hessian of l

$$\begin{aligned}\partial_{\theta_j}\partial_{\theta_i} l(f(\boldsymbol{\theta},\boldsymbol{x}),y) &= \partial_{\theta_j}\partial_{\theta_i} f(\boldsymbol{\theta},\boldsymbol{x}) l'(f(\boldsymbol{\theta},\boldsymbol{x})) \\ &+ \partial_{\theta_i} f(\boldsymbol{\theta},\boldsymbol{x})\partial_{\theta_j} f(\boldsymbol{\theta},\boldsymbol{x}) l''(f(\boldsymbol{\theta},\boldsymbol{x})),\end{aligned} \tag{6.7}$$

where l' (l'') is the first (second) derivative of the loss function with respect to the argument $f(\boldsymbol{\theta},\boldsymbol{x})$. In this chapter, we mostly use the loss function $l(f(\boldsymbol{\theta},\boldsymbol{x})) = 1-f(\boldsymbol{\theta},\boldsymbol{x})$, because $f(\boldsymbol{\theta},\boldsymbol{x})$ will be a figure of merit to be maximized, such as the overlap with some target state or a local observable. The only exception is the training on classical data where we use the square loss to compare the labels with an observable.

6.5 Behaviour of a General VQC without Data

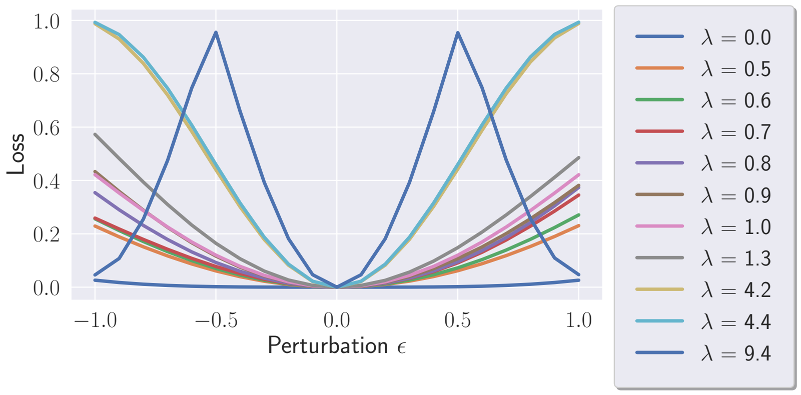


Figure 6.6: Loss around the mimimum for a given pertubation ϵ. We show several perturbations $\boldsymbol{\theta}_{\text{pert}} = \tilde{\boldsymbol{\theta}} + \epsilon\,\mathbf{v}$ labeled by the eigenvalue λ and the perturbation is along $\mathbf{v}$ which is the corresponding eigenvector of the eigenpair $(\lambda, \mathbf{v})$. Perturbations in the direction of a zero eigenvalue behave similarly and therefore we only show one as an example. Perturbations along the eigenvector of big eigenvalues lead to a much steeper increase in the loss.

We now focus on the behaviour of the eigenvalues of the Hessian during the training of a more general circuit with entangling gates. The aim of the training is, again, to optimize a randomly initialized circuit $V(\boldsymbol{\theta})$ such that we obtain the target state $|\psi_T\rangle = \sum_{\{\boldsymbol{\sigma}\}} |\boldsymbol{\sigma}\rangle$ at the output. To have a better circuit ansatz than in the previous section, we use general qubit rotations $R(\phi_1, \phi_2, \phi_3) = e^{i\phi_1\sigma_z} e^{i\phi_2\sigma_y} e^{i\phi_3\sigma_z}$, which we apply to each qubit, followed by a layer of controlled-Z (CZ) operations. We entangle the gates following the architecture proposed in Ref. [191], where after each layer of rotations we entangle each qubit with its neighbouring qubit, starting either with the even or the odd qubits. We define the layer $L_i(\boldsymbol{\theta}_i)$ as the sequence of rotating operations followed by CZ operations. Depending on the even/odd label i of the layer, we start the pairwise entanglement with the even/odd qubits [See Fig. 6.1 (b)].

Figure 6.5 shows the typical behaviour of the eigenvalues of the Hessian of such a parametrized circuit during the training. The eigenvalues' distribution for the randomly initialized circuit (at epoch 0) shows the occurrence of a Barren-Plateau-like behaviour with all eigenvalues close to zero. Already for a circuit with 4 qubits and 4 layers, which has 48 parameters (3 for each rotation), the circuit is likely to be initialized in a flat region.

In particular, Figure 6.5 (d) depicts the eigenvalues of the Hessian for the completely converged variational circuit (Epoch 15) and, similarly to the previous toy model, the eigenvalues are all non negative. Furthermore, most of the eigenvalues are zero, which means that the minimum is a flat pool, with only a few directions where the loss will increase and where most directions in the loss landscape are unaffected by small perturbations.

3D visualizations in these high dimensional loss landscapes are not feasible anymore, provided that it is not clear how to fix the remaining parameters. However, the Hessian contains information about the local curvature of the loss landscape. Considering an eigenvector $\mathbf{v_0}$ of the Hessian after convergence with the corresponding eigenvalue $\lambda_0 = 0$, we know that if we perturb these optimal parameters $\hat{\boldsymbol{\theta}}$ in the direction of $\mathbf{v_0}$, $\boldsymbol{\theta}_{\text{pert}} = \hat{\boldsymbol{\theta}} + \epsilon\,\mathbf{v_0}$, the loss will not change for a normalized $\mathbf{v_0}$ and a small perturbation $\epsilon \in \mathbb{R}$. The same idea applies to eigenvectors $\mathbf{v}_\lambda$ that correspond to an eigenvalue $\lambda \gg 0$. Perturbing $\hat{\boldsymbol{\theta}}$ in the direction of $\mathbf{v}_\lambda$ the loss increases fast, because the loss landscape in this direction is steep.

Figure 6.6 shows the perturbation in the direction of several eigenvectors. The corresponding eigenvalues are the labels of the curves. For the eigenvalue $\lambda = 0$ we see that the loss only increases for perturbations $\epsilon \gg 0$, which shows that the loss landscape is indeed flat around the minimum. We also would like to emphasize that most eigenvalues $\lambda = 0$, but we are only able to show the perturbation curve for one eigenpair $(\lambda_i = 0, \mathbf{v_i})$ in Fig. 6.6 because they behave very similarly around $\epsilon \ll 0$. The vast number of zero eigenvalues indicate that the loss landscape around the minimum is mostly flat with only a few steep directions.

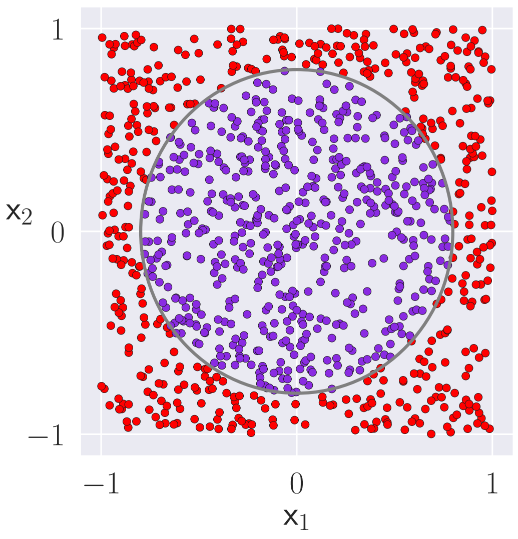


Figure 6.7: Training data for variational circuit. Data inside the disk are labelled −1 and data outside the disk are labelled +1.

We therefore see that the positive semi-definite nature of the Hessian is an indicator for a very stable solution. The optimization landscape cannot be perturbed and it is unlikely that during the optimization one jumps to another local minimum. The fact that in all directions the loss either stays the same or increases means that, with gradient descent methods the minimum loss can always be recovered, because we are in a locally convex point of the landscape. On the other hand, as we discuss in the next section, in quantum and classical ML tasks involving training with data not all eigenvalues are positive. There are still some small negative eigenvalues remaining and the Hessian is not positive semi-definite [223]. The optimization still converges, but the exact nature of the loss minima is controversial. There are claims that these flat regions are essential for the generalization capabilities of a NN [218] and that through these wide flat pools two solutions can be connected, which contradicts the idea of isolated basins at the bottom of the loss landscape [224].

6.6 Training with Data

We now characterize the Hessian of VQCs trained on data X with labels y, often called quantum neural networks (QNNs). We compare the results to a classical fully connected feed forward NN and we observe qualitative differences between classical NNs and QNNs. To this end, we use a simple dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ which only contains two classes, depicted in Fig. 6.7: a 2 dimensional set of points $\boldsymbol{x} = (x_1, x_2)$ labeled -1 inside the circle of radius r and 1 otherwise. In order to have a balanced dataset between the labels for $x_i \in [-1, 1]$, we choose $r = \sqrt{2/\pi}$.

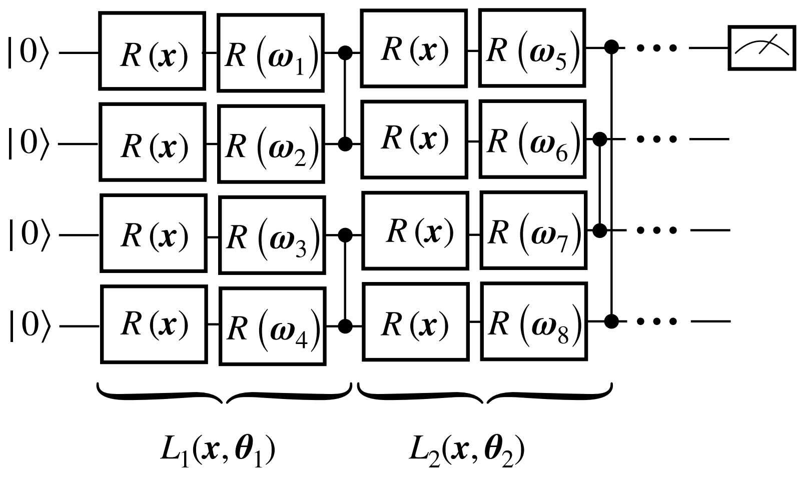


Figure 6.8: Data Reuploading Circuit. Each layer L_i contains an additional qubit rotation $R(\boldsymbol{x})$ with the classical data vector $\boldsymbol{x} = (x_1, x_2, x_3)$. The parameterization and the entangling gates are the same as in Figure 6.1 (b). For the 2D input data (x_1, x_2), we set $x_3 = 0$.

The first step is to encode the data instances $\boldsymbol{x} \in X$ onto the variational circuit and then the second step is to associate a certain measurement direction with a classical label y. There are many possible ways to encode the classical data in the quantum state $|\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle$. We follow here the architecture proposed in Ref. [191] and depicted in Fig. 6.8, which adds to each layer of Sec. 6.5 an additional rotation $R(x_1, x_2, x_3)$. This way, each layer consists of a data dependent rotation $R(x_1, x_2, x_3)$ followed by a parametrized rotation $R(\omega_1, \omega_2, \omega_3)$ applied on each qubit (we here choose $x_3 = 0$ as the data are two-dimensional), followed by the entangling operators. Regarding the label, the general idea is to define a measurement on one or more of the qubits that determines the QNN prediction. For two classes, the measurement of one qubit is already sufficient to have orthogonal measurements for each class, minimizing the overlap between complementary predictions. Therefore we choose to measure the first qubit in the Pauli Z direction and associate the measurement expectation value $\langle Z \rangle = 1$ ($\langle Z \rangle = -1$) with the labels $y = 1$ ($y = -1$). The loss $l(\langle Z \rangle, y)$ compares the label with the prediction $\langle Z \rangle$ and we here use the mean square loss

$$l(\boldsymbol{x}, y) = (\langle \psi(\boldsymbol{\theta}, \boldsymbol{x})| Z |\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle - y)^2. \tag{6.8}$$

After training, one finds the set of parameters $\hat{\boldsymbol{\theta}}$ that minimizes the empirical risk $\mathcal{L} = \sum_i l(\boldsymbol{x}_i, y_i)$.

To get an intuition, we first consider the Hessian of the loss of an arbitrary single training point $(\boldsymbol{x}, y)$. It has an eigenvalue distribution dependent on the prediction of the NN on $\boldsymbol{x}$. A correctly predicted label $\hat{y}$ with a loss $l(\boldsymbol{x}, \hat{y} = y)$ will lead to an eigenvalue distribution with most eigenvalues equal 0 and a few bigger than zero. This is the typical distribution if the loss of the NN is in a minimum [200]. For wrongly predicted labels, the eigenvalues of H are mostly zero with a few negative values. This is typical if the loss of the QNN is in a maximum. If the prediction $\hat{y}$ is somewhere between the labels -1 and 1 and there is a big uncertainty in the QNN's prediction, the eigenvalues of the Hessian will be again mostly zero but with an equal amount of positive and negative eigenvalues. This distribution is likely to be obtained in randomly initialized NNs. In the following, we show that for VQCs the eigenvalue distribution of the Hessian and the loss landscape itself shows some qualitative differences with respect to classical NNs. We also emphasize that we do not look at the Hessian spectrum of single training points, but rather at the spectrum of the whole training set. Empirical studies show that the eigenvalues of the sum of all the losses of the single training points (the empirical risk) behave approximately like the average of the spectrum of the single losses. If most training points are correctly classified, the Hessian of the empirical risk has negligible negative eigenvalues. In contrast, if all points are incorrectly classified the positive eigenvalues are negligible. If most training points show uncertainty in their prediction then there will be a a positive and a negative tail in the eigenvalue spectrum.

Figure 6.9 shows the prediction map of the QNN and the eigenvalues of the Hessian calculated over the whole training set (inset) for random initialization and after the training. We compare the results of the QNN with a classical fully connected feed forward NN (FFNN) in Figure 6.10. To make the results comparable, we choose a NN with a comparable amount of parameters, the same labels $y = \pm 1$ and the same loss function. The classical NN has 2 hidden layers with 12 and 10 neurons which results in 200 parameters, which is comparable to the 192 parameters of the QNN (8 qubits, with each 8 layers and 3 parameters per layer $8 \times 8 \times 3 = 192$).

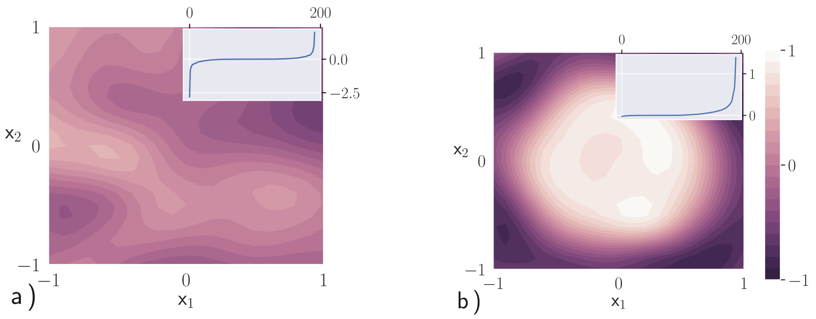


Figure 6.9: QNN label prediction map with Z-measurement for a randomly initialized variational circuit a) and after the training b). The inset shows the Hessian's eigenvalue distribution, which is similar to a classical NN if we use the loss function . The Hessian is computed over the whole training set.

The distribution of the eigenvalues is similar for the FFNN and the QNN. For a random initialization, the eigenvalues are equally likely positive and negative and the bulk is zero. After convergence, the negative eigenvalues disappear. There is a qualitative difference between the classical and the quantum version concerning the amplitude of the eigenvalues, which is almost an order of magnitude smaller for the FFNN for random initialization. For these relatively small FFNNs with 200 parameters, the distribution of the eigenvalues for a random initialization can vary strongly, but this is an effect that disappears for higher overparameterization of the FFNN. This is in contrast to the QNN that does not show strong fluctuations in the amplitude of the negative and positive eigenvalues.

Furthermore, the loss landscape of the QNN already shows steep slopes for random initialization compared to the flat random initialization of the FFNN. This seems to come from the fact that the loss of a QNN contains products and sums of sinusoidal functions like in Eq. (6.1), and stands in stark contrast to the training of variational circuits without data, where it is likely to initialize the circuit in absolutely flat regions, the Barren Plateaus. We did not observe any Plateaus in the case of QNN trained on data.

As the authors in Ref. [207] pointed out, classical NNs tend to transform local minima into deep and narrow steep valleys [225], which means that, for FFNN the amplitude of the negative eigenvalues tends to decrease if the training progresses. This is also in agreement with Ref. [202]. In addition, for FFNNs, we observe that there are just a few positive eigenvalues, of the order of the number of classes as introduced in section 6.1, in agreement with [200]. On the other side, QNNs do not seem to converge to these extremes, where just a few eigenvalues are positive and their positive "tail" in the eigenvalue spectrum is less steep (at least for the order of < 200 parameters). This observation seems to be connected with the fact that the loss landscape of the QNN after convergence still has some uncertainty at the decision boundary, unlike the FFNN which barely has any. The eigenvalues of the Hessian of the loss landscape inside a pool with high certainty are all close to zero, even if the NN prediction is wrong. This is generally true for wide flat regions of the loss landscape and it means that the main contribution of the eigenvalues of the Hessian comes from transition regions in the loss landscape where the label prediction changes from one to another.

6.7 Escaping from Barren Plateaus with the Help of the Hessian

In this section, we show how the Hessian can be used to escape from Barren Plateaus during the training of variational circuits. As shown in Ref. [204], the likelihood of initializing a variational circuit in a Barren Plateau increases with the numbers of qubits and layers of the circuit. Here, we propose to use the inverse of

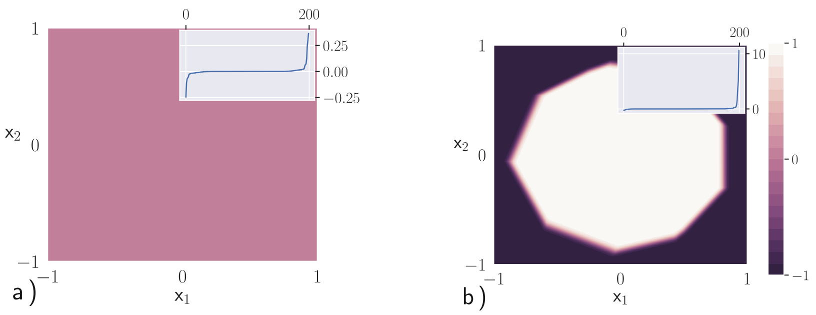


Figure 6.10: FFNN label prediction map for a randomly initialized NN a) and after the training b). The inset shows the Hessian's eigenvalue distribution. The Hessian is calculated over the whole training set.

the largest eigenvalue of the Hessian $1/\lambda_{\max}$ as the learning rate of the gradient descent optimizer in order to avoid being trapped in the loss landscape by vanishing gradients. Figure 6.11 shows a comparsion between a normal Gradient descent optimizer (GD), the quantum natural gradient optimizer (QNG) from Ref. [187] and our method of a learning rate of $1/\lambda_{\max}$. The comparison has been done for random initializations of a circuit with $N = 8$ qubits and 4 layers of parameterized rotations $R(\phi, \theta, \omega)$. The initial state is $|0\rangle^{\otimes N}$ and the target state is $|\psi_T\rangle = 1/\sqrt{2^N}\sum_{\boldsymbol{\sigma}}|\boldsymbol{\sigma}\rangle$. To compare the optimization methods we start in the same random initialization. We observe that the GD method is already struggling to train for such small circuits, because the gradients are too small, especially at the beginning of the training. With large learning rates, we might be able to escape the region of vanishing gradients, but it is impossible to converge. Both methods QNG and our Hessian method escape the flat region. The QNG method basically transforms the parameter update such that the update direction is not only in the steepest direction of the Euclidean space of the parameters, but in the steepest direction of the distribution space of all possible loss functions. Nevertheless, QNG can still get stuck in a flat region of the loss function. Hence, QNG provides poor performance whenever the circuit is initialized in a barren plateau. Our analysis also explains, why LBFGS optimization methods work better in ref. [191], provided that they are Hessian based.

Furthermore Ref. [207] shows that natural gradient optimizers help to prevent falling into local minima whereas Hessian

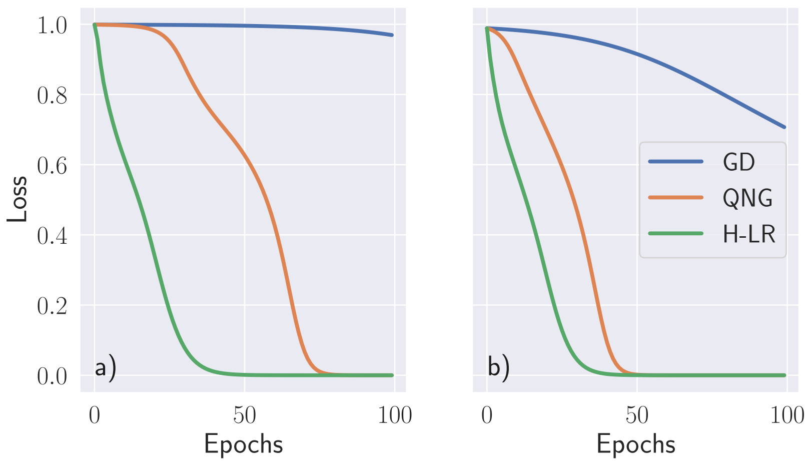


Figure 6.11: Training loss for training with Gradient descent (GD), Quantum Natural Gradient (QNG) and the learning rate adaption via the Hessian's maximum eigenvalue (H-LR) for two different random initializations a) and b). The first initialization a) shows a very slow convergence for GD and QNG in the beginning. This is caused by a Barren Plateau.

based methods struggle with local minima. The full Hessian is also costlier to calculate than the quantum metric tensor from Ref. [187]. Hence, we propose to use our method if the training is stuck in a flat region of the loss landscape to escape a Plateau, but to use the QNG for regions where the gradients are bigger. The two methods can, therefore, be combined: The QNG adapts the local shape of the gradient and fits it exactly to the loss landscape, providing faster convergence than vanilla GD [187] and the Hessian helps to kick the loss out of very flat regions. Finally, we emphasize that this method might have a practical shortcoming: as the eigenvalues of the Hessian are small in a Barren plateau, the measurement noise can be a limitation for such a proper evaluation of the Hessian in a plateau.

6.8 Conclusion

In this Chapter, we introduced the Hessian as a tool to study VQCs and discussed how the interpretation the eigenvalues and eigenvectors of the Hessian can lead to a better understanding of the loss landscape of a VQC. The combination of NISQ devices and gradient descent methods is still a young field and needs a thorough study for a better understanding of its potential. Especially for data driven tasks, the use of QML is at least controversial and one needs to find in which problems the application of QML has a potential advantage over classical NNs. We identified some qualitative differences between QNN and FFNN loss landscapes, which becomes evident after their initialization where the QNN loss shows a rougher surface than the FFNN which initializes very flat. We hope we can add another piece to the already controversial discussion of what kind of basins an ideal minima should live in for NNs by adding another interesting candidate: QNNs. Like in Chapter 5, the main bottleneck for the use of the Hessian in NN optimizations is its computational cost. For future research there is a need of finding good approximation schemes for quantum circuits. Furthermore, there are recent developments in classical ML showing that the learning rate early in SGD determines the quality of the minima found after the training [226] and big initial learning rates, like in our method with the inverse of the Hessian's biggest eigenvalue goes in this direction. Furthermore, there are Hessian based interpretability methods like the influence function [212, 227] which one could also apply to QNNs.

Chapter 7

Conclusions and Outlook

This thesis is dedicated to research at the intersection of ML and quantum physics. Two of the main topics that are studied in this context are the use of classical ML techniques to classify quantum states, in our case to distinguish phases of matter, and the use of quantum hardware to perform ML tasks. We have studied mainly three different directions within the realm of ML and quantum physics and quantum physics for ML. First, was the use of classical ML techniques to find phase transitions in quantum many-body systems. Second was how can we interpret these ML algorithms and their predictions to learn more about quantum systems and finally, how we can use methods that are inspired by classical ML interpretability to better understand quantum enhanced ML, also called QML. For the first two directions we focused on the use of deep NNs, which are very versatile function approximators. They learn to distill in a supervised and unsupervised manner patterns from quantum states to predict their phases of matter. One of the big disadvantages of NNs is that their reasoning is completely opaque and they, in general, do not allow us to gain insight of how a network came to a certain conclusion. Therefore, we used a machine learning interpretability method, called influence function, to better understand and unravel the black-box of NN predictions. Finally, inspired by the knowledge that we gained by classical ML interpretability, we study VQCs and their loss landscapes to understand how they compare to classical ML learning algorithms via the Hessian of the loss.

Domain Adversarial Phase Detection

In Chapter 3 we used a deep NN architecture, called domain adversarial NNs, to extract relevant features from quantum many-body ground states to successfully predict their phase. As humans, we often gain an intuition on a physical system using a special case that is analytically or numerically easy to treat. Then we generalize the insights to the more complex cases. Domain adaptation captures this idea: a deep learning system extracts intuition on a well-understood system and applies it to a more perplexing one. We benchmark the architecture first on the well known SSH model and see that the DANN predicts the exact phase transition points. Furthermore, we show that the DANN architecture outperforms traditional transfer learning when the target domain comes from a Hamiltonian with noisy interactions and it can predict new and unknown phases reliably. When we apply the DANN on the spin-1/2 Heisenberg chain that shows a MBL transition we see that the DANN is able to "invent" a new "order parameter" for the MBL phase transition that yields meaningful results from vastly fewer disorder realizations than established methods. This result is a strong argument for the use of ML techniques in noisy systems and indicates a clear advantage over standard methods. It seems fair to say that the resulting quantity actually captures the essential physics, as the network, once trained, can correctly identify the phase transition not only at different energy densities, but also in similar but distinct models. This is remarkable, since the MBL transition has mostly defied analytical approaches and even the question of what is the best way to delineate the phase could not be resolved in a satisfactory way.

We have demonstrated that ML can be used to automate the task of identifying relevant features that most efficiently capture the physics of phase transitions in quantum systems — a formidable task so far reserved for human researchers. We use the wavefunction as an input which is in our opinion the least biased information we could give a NN and we therefore reduce human input to a minimum. Our method is directly applicable to other nonstandard critical phenomena beyond SSH and MBL and can be used to distinguish multiple phases, even across different classes of models, as long as their Hilbert spaces are compatible [228]. It would be interesting for future work to apply the technique demonstrated here to larger system sizes, hope to gain new insights into whether there are additional thermodynamically stable phases near the MBL phase transition [229, 230]. To further improve our method one could combine it with the interpretability method introduced in Chapter 5 to gain a better understanding of what the NN learns. It would be important for this to try different architectures and to reduce the NN parameters to be able to calculate the Hessian.

Anomaly Detection

In Chapter 4 we reinterpret the phase prediction problem as an anomaly detection problem. The main idea behind this is that states or measurement data from the same phase of matter is similar and from different phases is different. This difference can be measured by the loss of a NN AE that was previously trained on states from one region of the phase diagram. The idea of using the similarity of state vectors that are close in phase space to identify phase transitions has already been used in the method of [136] but the use of our ML based method allows us to do the same without doing full state contraction, which is an enormous computational advantage. Compared to other ML techniques, the advantage of this method is that it is completely unsupervised and does not require any knowledge of the origin of the data or the underlying physical behavior. This method can be used to fully automate the exploration of unknown phase diagrams. We benchmark the method on the extended Bose-Hubbard model. As an input we use the entanglement spectrum (ES), the central tensor and correlators that could be observed via a measurement, all obtained by DMRG. We acknowledge that, compared to Chapter 3, this input data is in some sense more preprocessed than the wavefunction itself. At the same time avoiding the use of the full wavefunction allows for much better scalability. We cannot avoid human input completely because we have to make choices about the computational methods and e.g. the basis of the system. Nevertheless, we do not have to give the NN any additional information and we show that new phases can be found via the changes of the loss of the AE. Furthermore, we obtain a new region in the phase diagram of the extended Bose-Hubbard model that most likely shows a phase separation between the superfluid and super solid phase. Further research in this direction is still ongoing.

Moreover, there is ongoing research that aims at using anomaly detection on experimental data to further follow the idea of a self-driving laboratory, where the parameter space of a experimental setup can be scanned and directly fed into the AE to identify regions of interest.

Interpretability of NN phase prediction

In Chapter 5 we use ML interpretability to investigate what a NN learns in a supervised phase discrimination task. We use a method, called influence function, that approximates a leave one out (LOO) training. We train a CNN to classify the phase of ground states of the extended 1D half-filled spinless Fermi-Hubbard model. On the trained model we apply the influence function method to find out which states of the training set are most influential on the prediction of the NN of a test state. We provided strong evidence that the ML algorithm indeed learned a relevant order parameter describing the quantum phase transition. In simulations, where we don't teach the NN on the real order parameters, the values of the influence functions would still provide information about the phase transition and help in extracting a relevant order parameter. However it will not provide it explicitly. Moreover, we showed that through the influence functions, applied to the trained NN, we were able to detect an unknown phase. Two aspects impacted which training points were the most important for a given test point: The similarity with the test state and the uniqueness in the training data. Together they gave a notion of distance or similarity used by the CNN in the phase classification problem and indicated that the patterns relevant for the predictions coincided with the order parameters. We also acknowledge significant finite-size effects, but we see that the achieved results do not depend on the system size.

In future studies we can address open problems of topological models and MBL with NNs, whose logic can be finally discovered by influence functions. Even though this work concerned quantum phase transitions, this method may be easily applied to classical models as well as any physical model in general. Furthermore, it will be of interest to test this method on new architectures, such as in Chapter 3 and 4. For this we will have to use approximate methods of how to calculate the Hessian because the NN in the previous chapters have to many parameters to do exact calculations in reasonable time. Another way would be to use global average pooling (GAP) to reduce the size of known convolutional architectures to make the calculation of the Hessian feasible. Furthermore, it would be interesting to use different input data, e.g. the ES like in Chapter 4 and see how the influence function would behave when reducing finite size effects. There are as well ongoing studies of how we can use the influence function on experimental data. Moreover, this tool proved to be very sensitive to outliers existing in the data set and may serve for anomaly detection. As such, it can be useful for analysis of experimental noise in the data on which various models are built and allow to judge how strongly it affects them.

Loss Landscapes of Variational Quantum Circuits

The study of the Hessian of general variational methods is a useful tool to gain an understanding of the shape of the loss landscape. As shown in Chapter 5, the Hessian can even help us to find correlations between test and training examples and express this correlation as an influence function. For the study of VQCs the questions are still more fundamental, because it is not yet clear if there is any advantage in the use of quantum circuits for ML applications, at least on NISQ devices. Our work contributes a tool that allows us to better study VQCs that can help to identify possible differences of VQCs to classical ML algorithms that might be a resource for a quantum advantage. We first show on a simple toy example how the eigenvalues of the Hessian can be interpreted and what information can be extracted from them. Then we introduce a way to calculate the Hessian on an actual quantum hardware by applying the parameter shift rule twice. We apply the Hessian to a bigger VQC and show how it can help to escape a Barren Plateau. Furthermore, we study the Hessian of a VQC trained on data, a QNN. Classical NN are useful because of many reasons, but one of their main advantages is the fact that they can be overparametrized in a highly non-linear way. It is assumed that overparametrization leads to the connection of local minima in loss landscapes up to the complete disappearance of local minima traps. In Chapter 6 we show evidence that the the loss minimas of QNNs behave similarly to classical NNs if we compare circuits with the same amount of parameters. A well converged QNN and VQC loss landscape shows eigenvalues of the

Hessian that indicate that the minimum is a mostly flat pool with just a few steep directions which is in agreement what we find in classical NN. We as well identified some qualitative differences between QNN and classical NN loss when we look at the loss landscape after initialization. In the training data input space the QNN loss shows a rougher surface than the FFNN which initializes very flat. This is very likely because of the way the data is embedded in the QNN. Because the data is repeatedly uploaded to the QNN, a change in the input data is equivalent to a collective change of many parameters at the same time and, therefore, the output changes rapidly.

For future research this feature could point in the direction of how to lead a QNN out of a Plateau. If the training is stuck one could apply changes to a collection of parameters instead of single parameter updates. For further future research in the context of the Hessian based analysis of loss landscapes in QML it will be crucial to develop tools to approximate the Hessian more efficiently. In this thesis we calculate the exact Hessian of a loss landscape which is not necessary in many applications e.g. escaping Barren Plateaus can be done with a very rough estimate of the Hessian. Furthermore, it would be interesting to study the influence function from Chapter 5 in QNNs. And finally, we would like to point out that there are striking similarities between the minimization of VQCs and Quantum Monte Carlo. In particular, in both scenarios, it seems advantageous to take into account the local curvature through the (quantum) metric tensor or the Hessian. Thus, it would be interesting to explore the connections between these two fields.

Appendix A

DANN: Data analysis MBL

For the Heisenberg model we analyze the data generated by the NN in the way described in the main text. In particular, to obtain an estimate of the energy resolved phase boundary in the thermodynamic limit, we compute the output of the phase classifier as a function of h for various values of ϵ and then perform a data collapse as described in the main text. The raw and collapsed data for different values of ϵ is shown in Figure 3.9. The resulting estimates for h, ν, and their errors are shown in Table A.1.

ϵ	$\nu \pm \Delta\nu$	$h_c \pm \Delta h_c$
0.2	1.5 ± 0.2	1.8 ± 0.4
0.3	1.6 ± 0.2	2.4 ± 0.4
0.4	1.6 ± 0.2	3.0 ± 0.2
0.5	1.6 ± 0.1	3.5 ± 0.2
0.6	1.5 ± 0.2	3.0 ± 0.2
0.7	1.6 ± 0.2	2.5 ± 0.4
0.8	1.6 ± 0.2	2.2 ± 0.4

Table A.1: Estimates for ν and h_c as well as their errors. The errors were conservatively estimated by fixing the best possible value for the respective other quantity and determining from plots such as those in Figure 3.9 when the data collapse would diverge visibly.

Appendix B

Anomaly Detection: Phase discovery

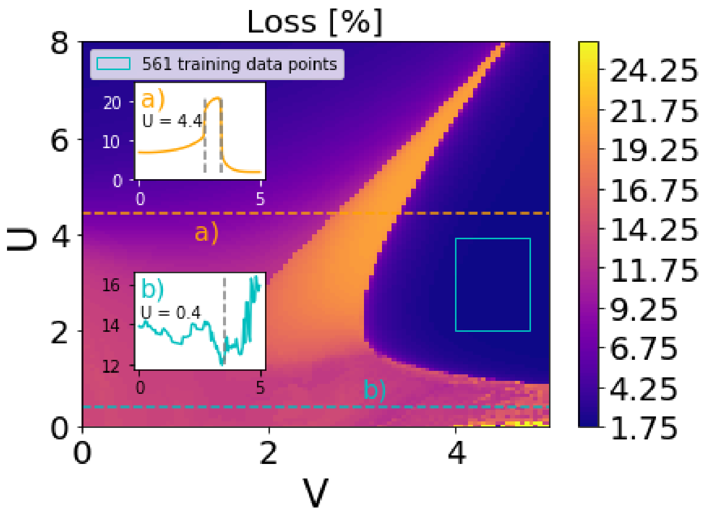


Figure B.1: 2D loss map of the AE after training with ES from blue square frame. The insets a), b) show the loss along the dashed lines. The training in the region of high loss from fig. 3 in the main text confirms the boundaries. The MI and HI phases are well separated because the loss in HI is much higher in comparison (inset a)). This isn't necessarily the case as indicated in inset b) for SF and SF+SS.

As discussed in the main text, it is not necessarily always the case that after one training iteration, one can differentiate the different phases inside the high-loss anomalous region, as was the case in Fig. 4.5 in the main text. Thus, we propose picking homogeneous and high contrast anomalous regions after the initial training as a systematic approach. In B.1 we do this for $(U, V) \in [4, 4.8] \times [2, 4]$, which is the region where the loss after the initial training was highest and accounts to the DW phase. We can confirm the previously determined boundaries to the anomalous region, which is very sharp due to the Ising type transition. Further, we can again separate MI and HI due to different loss levels but without a valley in between (B.1, inset a)).

Appendix C

Anomaly Detection: Phases of the extended Bose Hubbard model

We briefly summarize the phases of the extended Bose Hubbard model at integer filling, that have not been discussed in the main text.

MI-HI-DW transition

Another way to locate the phase transitions, as has been discussed in the main text, is by looking at the entanglement entropy

$$S^{[i]} = -\sum_v \Lambda_v^{[i]2} \log_2(\Lambda_v^{[i]2}), \tag{C.1}$$

and the correlation length ξ, defined in terms of

$$\mu_2 = \exp(-L/\xi), \tag{C.2}$$

where μ_2 is the second largest eigenvalue of the transfer matrix of the L-site unit cell. Note that here all entanglement entropies are equal along the chain, such that we will simply speak of S. We fix $U = 5$ and compare these two quantities with the order parameters in C.1. We note that due to the infinite nature of the states, ξ and S were not susceptible to L, only to χ_{max}. Since the order parameters do not notably change with χ_{max}, we fix $\chi_{\text{max}} = 100$, for which S and ξ indicate well the transitions.

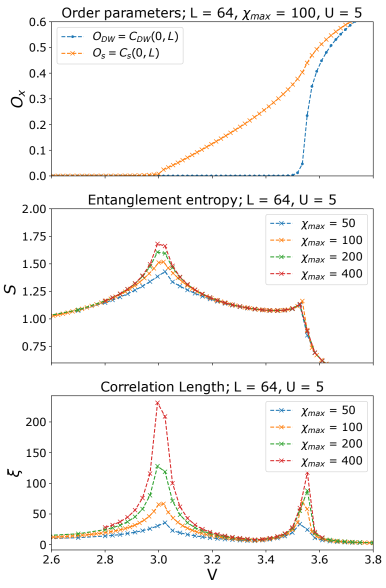


Figure C.1: Mott-Insulator to Haldane-Insulator to Density Wave transition characterized by a) order parameters (correlators evaluated at $r = L$), b) Entanglement entropy and b) correlation length to determine the phase transition, exemplarily for $U = 5$.

As mentioned in the main text, another way to indicate quantum phase transitions without any a priori knowledge is by calculating overlaps (fidelity) between ground states in the phase diagram [136],

$$\mathcal{F}(V_i, V_j) = \langle \Psi(V_i) | \Psi(V_j) \rangle \,. \tag{C.3}$$

For example, we depict the MI-HI-DW transition for $U = 5$ in C.2. We can see how the states in the different phases separate into quasi non-overlapping regions. The off-diagonal $\mathcal{F}(V_i, V_{i+1})$ accurately yields the transition points, indicated by drops in fidelity [136].

Critical Superfluid phase

In the superfluid phase, long-range order accounts to a power-law decay in C_{SF}, which gradually turns into an exponential decay in the transition. We graphically display this in C.3 by showing the decay and a respective fit deep in the SF and MI phases. Near the transition point ($U = 4.1$ in C.3) we can clearly see that it

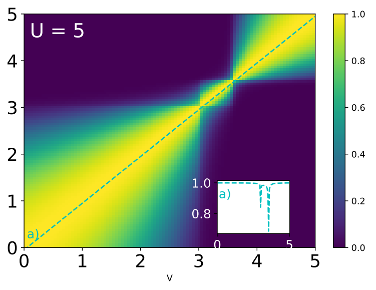


Figure C.2: $\mathcal{F}(V_i, V_j)$ for all possible combinations on 100 equally spaced $V_i \in [0, 5]$ for fixed $U = 5$. Inset: Off diagonal indicating transition points with drops in overlap.

is neither power-law nor exponential decay. Although it nicely illustrates the quality of the transition, it is hard to determine the exact transition point in this manner.

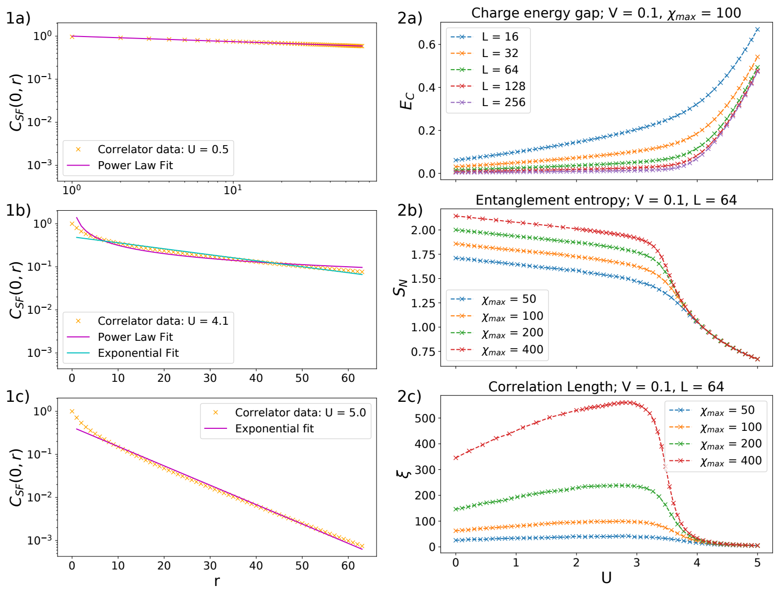


Figure C.3: 1) Correlators for the superfluid phase for fixed nearest neighbour interaction $V = 0.1$ 1a) deep in SF phase, 1b) near the transition point and 1c) deep in MI phase. 2) Superfluid to Mott-Insulator transition characterized 2a) by vanishing charge energy gap, 2b) diverging entanglement entropy and 2c) diverging correlation length.

Another indicator for this transition is the closing charge energy gap

$$E_C = E(L+1) + E(L-1) - 2E(L), \tag{C.4}$$

where $E(n)$ denotes the ground state energy for n fixed bosons. The superfluidity of the state is indicated by zero E_C, which we gradually approach in the thermodynamic limit, see C.3. The correlation length ξ and entanglement entropy S_N diverge in the critical SF phase, as indicated in C.3. We see that it is hard to estimate the transition point from all these physical quantities. The best results are obtained from looking at the overlaps, C.3, in C.4. Though, the contrast in the off-diagonal as indicated in the inset is very low. Note that the criticality of SF leads to the valleys between MI, HI and DW, depicted in fig. 3 in the main text.

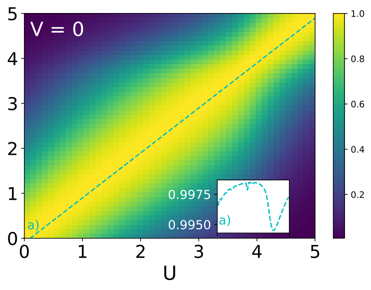


Figure C.4: $\mathcal{F}(U_i, U_j)$ for all possible combinations on 100 equally spaced $U_i \in [0, 5]$ for fixed $V = 0$. Inset: Off diagonal indicating transition point with drops in overlap. Note the low contrast.

Bibliography

[1] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, 2007.

[2] L. Ward and C. Wolverton. Atomistic calculations and materials informatics: A review. *Curr. Opin. Solid State Mater. Sci.*, 21:167, 2016.

[3] E. M. Christiansen *et al.* *In silico* labeling: Predicting fluorescent labels in unlabeled images. *Cell*, 173:792, 2018.

[4] D. Wong and S. Yip. Machine learning classifies cancer. *Nature*, 555:446, 2018.

[5] B. Naul, J. S. Bloom, F. Pérez, and S. van der Walt. A recurrent neural network for classification of unevenly sampled variable stars. *Nat. Astron.*, 2:151–155, 2018.

[6] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nat. Commun.*, 5:4308, 2014.

[7] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo. Neural-network quantum state tomography. *Nat. Phys.*, 14:447, 2018.

[8] J Carrasquilla, G. Torlai, and R. G. Melko. Latent space purification via neural density operators. *Nat. Mach. Intell.*, 1:155–161, 2019.

[9] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta. Reinforcement learning in different phases of quantum control. *Phys. Rev. X*, 8:031086, 2018.

[10] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nat. Phys.*, 13:431–434, 2017.

[11] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber. Learning phase transitions by confusion. *Nat. Phys.*, 13:435–439, 2017.

[12] Frank Schäfer and Niels Lörch. Vector field divergence of predictive model output as indication of phase transitions. *Phys. Rev. E*, 99:062107, Jun 2019.

[13] A. Tanaka and A. Tomiya. Detection of phase transition via convolutional neural networks. *J. Phys. Soc. Jpn.*, 86:063001, 2017.

[14] C.-D. Li, D.-R. Tan, and F.-J. Jiang. Applications of neural networks to the studies of phase transitions of two-dimensional Potts models. *Ann. Phys.*, 391:312, 2018.

[15] Ye-Hua Liu and Evert P. L. van Nieuwenburg. Discriminative Cooperative Networks for Detecting Phase Transitions. *Physical Review Letters*, 120(17):176401, April 2018.

[16] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Sci. Rep.*, 7:8823, 2017.

[17] Kelvin Ch'ng, Nick Vazquez, and Ehsan Khatami. Unsupervised machine learning account of magnetic transitions in the hubbard model. *Phys. Rev. E*, 97:013306, Jan 2018.

[18] H. Théveniaut and F. Alet. Neural network setups for a precise detection of the many-body localization transition: finite-size scaling and limitations. *Phys. Rev. B*, 100:224202, 2019.

[19] S. J. Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E*, 96:022140, 2017.

[20] D.-L. Deng, X. Li, and S. Das Sarma. Quantum entanglement in neural network states. *Phys. Rev. X*, 7:021021, 2017.

[21] P. Zhang, H. Shen, and H. Zhai. Machine learning topological invariants with neural networks. *Phys. Rev. Lett.*, 120:066401, 2018.

[22] Yuan-Hong Tsai, Meng-Jer Yu, Yu-Hao Hsu, and Ming-Chiang Chung. Deep learning of topological phase transitions from entanglement aspects. 2019.

[23] Eliska Greplova, Agnes Valenti, Gregor Boschung, Frank Schäfer, Niels Lörch, and Sebastian Huber. Unsupervised identification of topological order using predictive models. 2019.

[24] P. Ponte and R. G. Melko. Kernel methods for interpretable machine learning of order parameters. *Phys. Rev. B*, 96:205146, 2017.

[25] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, October 2009.

[26] X.-D. Cai, Christian Weedbrook, Z.-E. Su, M.-C. Chen, Mile Gu, M.-J. Zhu, L. Li, N.-L. Liu, Chao-Yang Lu, and Jian-Wei Pan. Experimental Quantum Computing to Solve Systems of Linear Equations. *Physical Review Letters*, 110(23):230501, June 2013.

[27] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[28] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv:1307.0411 [quant-ph]*, November 2013.

[29] 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, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David

Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019.

[30] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, September 2017.

[31] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *arXiv preprint arXiv:2010.02174*, 2020.

[32] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, September 2014.

[33] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, September 2014.

[34] Hartmut Neven, Vasil S. Denchev, Geordie Rose, and William G. Macready. Training a Large Scale Classifier with the Quantum Adiabatic Algorithm. *arXiv:0912.0779 [quant-ph]*, December 2009.

[35] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas

P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119(19):10856–10915, October 2019.

[36] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014.

[37] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits. *arXiv:1904.04767 [quant-ph]*, April 2019.

[38] Ryan Sweke, Jean-Pierre Seifert, Dominik Hangleiter, and Jens Eisert. On the quantum versus classical learnability of discrete distributions. *arXiv preprint arXiv:2007.14451*, 2020.

[39] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, February 2020.

[40] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *arXiv:2008.08605 [quant-ph, stat]*, August 2020.

[41] Xiao-Yu Dong, Frank Pollmann, and Xue-Feng Zhang. Machine learning of quantum phase transitions. *Phys. Rev. B*, 99:121104, Mar 2019.

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

[43] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Fläschner, C. Becker, K. Sengstock, and C. Weitenberg. Identifying quantum phase transitions using artificial neural networks on experimental data. *Nat. Phys.*, 15:917–920, 2019.

[44] E. Khatami, E. Guardado-Sanchez, B. M. Spar, J. F. Carrasquilla, W. S. Bakr, and R. T. Scalettar. Visualizing correlations in the 2d fermi-hubbard model with ai. *arXiv:2002.12310*, 2020.

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

[46] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[47] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, pages 1–13, 2017.

[48] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9428–9433, 2019.

[49] W. Zhang, L. Wang, and Z. Wang. Interpretable machine learning study of the many-body localization transition in disordered quantum Ising spin chains. *Phys. Rev. B*, 99:054208, 2019.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, November 2016.

[51] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *arXiv:1911.02685 [cs, stat]*, June 2020.

[52] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani. *Advances in Domain Adaptation Theory.* Elsevier Science, 2019.

[53] Ievgen Redko, Amaury Habrard, Emilie Morvant, Marc Sebban, and Younès Bennani. Domain Adaptation Problem. In *Advances in Domain Adaption Theory*, pages 21–36. Elsevier, 2019.

[54] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *arXiv:1505.07818 [cs, stat]*, May 2016.

[55] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[56] Qing Liu, Ningyu Zhang, Wenzhu Yang, Sile Wang, Zhenchao Cui, Xiangyang Chen, and Liping Chen. A Review of Image Recognition with Deep Convolutional Neural Network. In De-Shuang Huang, Vitoantonio Bevilacqua, Prashan Premaratne, and Phalguni Gupta, editors, *Intelligent Computing Theories and Application*, Lecture Notes in Computer Science, pages 69–80, Cham, 2017. Springer International Publishing.

[57] Iroju Olaronke and J. Olaleke. A Systematic Review of Natural Language Processing in Healthcare. *International Journal of Information Technology and Computer Science*, 08:44–50, August 2015.

[58] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep Learning Techniques for Music Generation – A Survey. *arXiv:1709.01620 [cs]*, August 2019.

[59] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[60] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[61] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, 9(4):611–629, August 2018.

[62] Ajay Agrawal, Joshua Gans, and Avi Goldfarb. *Prediction Machines: The Simple Economics of Artificial Intelligence.* Harvard Business Review Press, Boston, MA, USA, 2018.

[63] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[64] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[65] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[66] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pages 2280–2288, 2016.

[67] Christoph Molnar. *Interpretable Machine Learning.* 2019. https://christophm.github.io/interpretable-ml-book/.

[68] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[69] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[70] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[71] Philip Adler, Casey Falk, Sorelle A Friedler, Tionney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 54(1):95–122, 2018.

[72] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[73] R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19:15–18, February 1977.

[74] R. D. Cook and S. Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

[75] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression.* Chapman and Hall, New York and London, 1982.

[76] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894, 2017.

[77] P. W. Koh, K.-S. Ang, H. H. K. Teo, and P. Liang. On the accuracy of influence functions for measuring group effects. *arXiv:1905.13289*, 2019.

[78] H. Philathong, V. Akshay, I. Zacharov, and J. Biamonte. Computational Phase Transition Signature in Gibbs Sampling. *arXiv:1906.10705 [cond-mat, physics:quant-ph]*, June 2019.

[79] Andrew Canning and Jean-Pierre Naef. Phase diagrams and the instability of the spin glass states for the diluted Hopfield neural network model. *Journal de Physique I*, 2(9):1791–1801, 1992.

[80] L. Landau. The Theory of Phase Transitions. *Nature*, 138(3498):840–841, November 1936.

[81] S. Sachdev. *Quantum Phase Transitions.* Cambridge University Press, 2011.

[82] Florian Häse, Loïc M. Roch, and Alán Aspuru-Guzik. Next-Generation Experimentation with Self-Driving Laboratories. *Trends in Chemistry*, 1(3):282–291, jun 2019.

[83] David J. Luitz, Nicolas Laflorencie, and Fabien Alet. Many-body localization edge in the random-field Heisenberg chain. *arXiv:1411.0660 [cond-mat]*, December 2014.

[84] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]*, August 2016.

[85] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

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

[87] Alan Morningstar and Roger G. Melko. Deep learning the Ising model near criticality. August 2017.

[88] David Ceperley and Berni Alder. Quantum Monte Carlo. *Science*, 231(4738):555–560, February 1986.

[89] J. Carlson, S. Gandolfi, F. Pederiva, Steven C. Pieper, R. Schiavilla, K. E. Schmidt, and R. B. Wiringa. Quantum Monte Carlo methods for nuclear physics. *Reviews of Modern Physics*, 87(3):1067–1118, September 2015.

[90] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. Quantum Monte Carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33–83, January 2001.

[91] Stefan Rommer. Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group. . . ., page 18.

[92] Steven R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, November 1992.

[93] Ulrich Schollwoeck. The density-matrix renormalization group in the age of matrix product states. *arXiv:1008.3477 [cond-mat]*, January 2011.

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

[95] F. Verstraete, V. Murg, and J. I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, March 2008.

[96] Giuseppe Carleo and Matthias Troyer. Solving the Quantum Many-Body Problem with Artificial Neural Networks. *Science*, 355(6325):602–606, February 2017.

[97] Giacomo Torlai and Roger G. Melko. Learning Thermodynamics with Boltzmann Machines. *Physical Review B*, 94(16):165134, October 2016.

[98] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, November 1994.

[99] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv:quant-ph/9605043*, November 1996.

[100] P. B. M. Sousa and R. V. Ramos. Universal quantum circuit for n-qubit quantum gate: A programmable quantum gate. *arXiv:quant-ph/0602174*, May 2006.

[101] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. The Quantum Approximation Optimization Algorithm for MaxCut: A Fermionic View. *Physical Review A*, 97(2):022304, February 2018.

[102] Francesco Tacchino, Panagiotis Barkoutsos, Chiara Macchiavello, Ivano Tavernelli, Dario Gerace, and Daniele Bajoni. Quantum implementation of an artificial feed-forward neural network. *Quantum Science and Technology*, 5(4):044010, October 2020.

[103] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *arXiv:2008.08605 [quant-ph, stat]*, August 2020.

[104] Philip Warren Anderson. Absence of diffusion in certain random lattices. *Physical Review*, 109:1492, Mar 1958.

[105] Serge Aubry and Gilles André. Analyticity breaking and Anderson localization in incommensurate lattices. *Ann. Israel Phys. Soc*, 3(133):18, 1980.

[106] Jian Li, Rui-Lin Chu, J. K. Jain, and Shun-Qing Shen. Topological Anderson insulator. *Physical Review Letters*, 102:136806, Apr 2009.

[107] Ian Mondragon-Shem, Taylor L. Hughes, Juntao Song, and Emil Prodan. Topological criticality in the chiral-symmetric AIII class at strong disorder. *Physical Review Letters*, 113:046802, 2014.

[108] Rahul Nandkishore and David A. Huse. Many-body localization and thermalization in quantum statistical mechanics. *Annual Review of Condensed Matter Physics*, 6(1):15, mar 2015.

[109] 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. *Ann. Phys. (N. Y).*, 321(5):1126–1205, may 2006.

[110] Arijeet Pal and David A. Huse. Many-body localization phase transition. *Physical Review B*, 82(17):174411, November 2010.

[111] Rahul Nandkishore and David A Huse. Many-Body Localization and Thermalization in Quantum Statistical Mechanics. *Annu. Rev. Condens. Matter Phys.*, 6(1):15–38, mar 2015.

[112] David J. Luitz, Nicolas Laflorencie, and Fabien Alet. Many-body localization edge in the random-field Heisenberg chain. *Phys. Rev. B*, 91(8), 2015.

[113] Thimothée Thiery, François Huveneers, Markus Müller, and Wojciech De Roeck. Many-body delocalization as a quantum avalanche. pages 1–9, jun 2017.

[114] S. A. Parameswaran and Romain Vasseur. Many-body localization, symmetry, and topology. jan 2018.

[115] Francesca Pietracaprina, Nicolas Macé, David J. Luitz, and Fabien Alet. Shift-invert diagonalization of large many-body localizing spin chains. mar 2018.

[116] Dmitry A. Abanin, Ehud Altman, Immanuel Bloch, and Maksym Serbyn. Many-body localization, thermalization, and entanglement. *Reviews of Modern Physics*, 91(2):021001, May 2019.

[117] 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, September 2013.

[118] David A. Huse, Rahul Nandkishore, and Vadim Oganesyan. Phenomenology of fully many-body-localized systems. *Physical Review B*, 90(17):174202, November 2014.

[119] T. Enss, F. Andraschko, and J. Sirker. Many-body localization in infinite chains. *Phys. Rev. B*, 95(4):045121, jan 2017.

[120] Kazue Kudo and Tetsuo Deguchi. Finite-Size Scaling Regarding Interaction in the Many-Body Localization Transition. mar 2018.

[121] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[122] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(85):2579–2605, 2008.

[123] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 2016.

[124] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. *Biometrics*, 2002.

[125] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of SIGKDD-96, 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, August 1996.

[126] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[127] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. PatrickHuembeli/Adversarial-Domain-Adaptation-for-Identifying-Phase-Transitions: DANN_Arxiv_Version_01, October 2017. `https://github.com/PatrickHuembeli/Adversarial-Domain-Adaptation-for-Identifying-Phase-Transitions`.

[128] János K. Asbóth, László Oroszlány, and András Pályi. *A Short Course on Topological Insulators*, volume 919 of *Lecture Notes in Physics*. Springer International Publishing, 2016.

[129] Navketan Batra and Goutam Sheet. Understanding Basic Concepts of Topological Insulators Through Su-Schrieffer-Heeger (SSH) Model. *Resonance*, 25(6):765–786, June 2020.

[130] Maria Maffei, Alexandre Dauphin, Filippo Cardano, Maciej Lewenstein, and Pietro Massignan. Topological characterization of chiral models through their long time dynamics. 2017.

[131] Fabien Alet and Nicolas Laflorencie. Many-body localization: An introduction and selected topics. *arXiv:1711.03145 [cond-mat]*, May 2018.

[132] Luca D'Alessio, Yariv Kafri, Anatoli Polkovnikov, and Marcos Rigol. From quantum chaos and eigenstate thermalization to statistical mechanics and thermodynamics. *Advances in Physics*, 65(3):239–362, May 2016.

[133] A. B. Harris. Effect of random defects on the critical behaviour of Ising models. *Journal of Physics C: Solid State Physics*, 7(9):1671, May 1974.

[134] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. 2017.

[135] Guillermo Valle-Pérez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. 2018.

[136] P. Zanardi, M. Cozzini, and P. Giorda. Ground state fidelity and quantum phase transitions in free Fermi systems. jun 2006.

[137] Roman Orus. A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. jun 2013.

[138] Xiaolong Deng and Luis Santos. Entanglement spectrum of one-dimensional extended Bose-Hubbard models. apr 2011.

[139] Kazuya Shinjo, Kakeru Sasaki, Satoru Hase, Shigetoshi Sota, Satoshi Ejima, Seiji Yunoki, and Takami Tohyama. Machine Learning Phase Diagram in the Half-filled One-dimensional Extended Hubbard Model. apr 2019.

[140] Yuan-Hong Tsai, Meng-Jer Yu, Yu-Hao Hsu, and Ming-Chiang Chung. Deep learning of topological phase transitions from entanglement aspects. sep 2019.

[141] Jianfeng Dong, Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Learning Deep Representations Using Convolutional Auto-encoders with Symmetric Skip Connections. nov 2016.

[142] Jianfeng Dong, Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Unsupervised feature learning with symmetrically connected convolutional denoising auto-encoders. *CoRR*, abs/1611.09119, 2016.

[143] Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner. Discovering Physical Concepts with Neural Networks. *Physical Review Letters*, 124(1):010508, jan 2020.

[144] 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.

[145] Korbinian Kottmann and Patrick Huembeli. Unsupervised phase discovery with deep anomaly detection. https://github.com/Qottmann/phase-discovery-anomaly-detection/, 2020.

[146] Davide Rossini and Rosario Fazio. Phase diagram of the extended Bose Hubbard model. apr 2012.

[147] Till D. Kuehner and H. Monien. Phases of the one-dimensional Bose-Hubbard model. dec 1997.

[148] Till D. Kuehner, Steven R. White, and H. Monien. The one-dimensional Bose-Hubbard Model with nearest-neighbor interaction. jun 1999.

[149] Tapan Mishra, Ramesh V. Pai, S. Ramanan, Meetu Sethi Luthra, and B. P. Das. Supersolid and solitonic phases in one-dimensional Extended Bose-Hubbard model. jul 2009.

[150] Laura Urba, Emil Lundh, and Anders Rosengren. One-dimensional extended Bose-Hubbard model with a confining potential: A DMRG analysis. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 39(24):5187–5198, jul 2006.

[151] Satoshi Ejima, Florian Lange, and Holger Fehske. Spectral and Entanglement Properties of the Bosonic Haldane Insulator. jul 2014.

[152] M. A. Cazalilla, R. Citro, T. Giamarchi, E. Orignac, and M. Rigol. One dimensional Bosons: From Condensed Matter Systems to Ultracold Gases. jan 2011.

[153] G. G. Batrouni, F. Hebert, and R. T. Scalettar. Supersolid phases in the one dimensional extended soft core Bosonic Hubbard model. may 2006.

[154] Erez Berg, Emanuele G. Dalla Torre, Thierry Giamarchi, and Ehud Altman. Rise and fall of hidden string order of lattice bosons. mar 2008.

[155] Pietro Silvi, Ferdinand Tschirsich, Matthias Gerster, Johannes Jünemann, Daniel Jaschke, Matteo Rizzi, and Simone Montangero. The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems. *SciPost Phys. Lect. Notes*, page 8, 2019.

[156] Johannes Hauschild and Frank Pollmann. Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy). *SciPost Phys. Lect. Notes*, page 5, 2018. Code available from `https://github.com/tenpy/tenpy`.

[157] Marcel den Nijs and Koos Rommelse. Preroughening transitions in crystal surfaces and valence-bond phases in quantum spin chains. *Physical Review B*, 40(7):4709–4734, September 1989.

[158] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[159] Guifre Vidal. Efficient classical simulation of slightly entangled quantum computations. jan 2003.

[160] G. Vidal. Classical simulation of infinite-size quantum lattice systems in one spatial dimension. may 2006.

[161] Ian P. McCulloch. From density-matrix renormalization group to matrix product states. jan 2007.

[162] R. Orús and G. Vidal. Infinite time-evolving block decimation algorithm beyond unitary evolution. *Physical Review B*, 78(15):155117, oct 2008.

[163] Keima Kawaki, Yoshihito Kuno, and Ikuo Ichinose. Phase diagrams of the extended bose-hubbard model in one dimension by monte-carlo simulation with the help of a stochastic-series expansion. *Phys. Rev. B*, 95(19):195101, May 2017.

[164] Yung-Chung Chen and Min-Fong Yang. Two supersolid phases in hard-core extended Bose-Hubbard model. mar 2017.

[165] Y. Zhang and E.-A. Kim. Quantum loop topography for machine learning. *Phys. Rev. Lett.*, 118:216401, May 2017.

[166] M. J. S. Beach, A. Golubeva, and R. G. Melko. Machine learning vortices at the Kosterlitz-Thouless transition. *Phys. Rev. B*, 97:045207, 2018.

[167] M. Richter-Laskowska, H. Khan, N. Trivedi, and M. M. Maśka. A machine learning approach to the berezinskii-kosterlitz-thouless transition in classical and quantum models. *Condens. Matter Phys.*, 21(3):33602, 2018.

[168] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. *arXiv:1802.01933*, 2018.

[169] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

[170] A. Baehrens, T. Schroeter, and S. Harmeling. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, 2010.

[171] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you? Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[172] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[173] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *arXiv:1610.02391*, 2016.

[174] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-CAM++: Improved visual explanations for deep convolutional networks. In *IEEE Winter Conf. on Applications of Computer Vision (WACV2018)*, 2018.

[175] Q. Zhang and S.-C. Zhu. Visual interpretability for deep learning: a survey. *Front. Inform. Technol. Electron. Eng.*, 19:27–39, 2018.

[176] EU General Data Protection Regulation (GDPR). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ*, L 119/1, 2016.

[177] T. Songül. Fair and unbiased algorithmic decision making. *JRC Tech. Rep.*, 10, 2018.

[178] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representation*, 2017.

[179] W. Brendel and M. Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet. In *International Conference on Learning Representations*, 2019.

[180] A Dawid, P. Huembeli, M. Tomza, M. Lewenstein, and A. Dauphin. `http://doi.org/10.5281/zenodo.3746540`, 2020. GitHub repository: Interpretable-Phase-Classification (Version arXiv1.0).

[181] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 2013.

[182] O. Dutta, M. Gajda, P. Hauke, M. Lewenstein, D.-S. Lühmann, B. A. Malomed, T. Sowiński, and J. Zakrzewski. Non-standard Hubbard models in optical lattices: a review. *Rep. Prog. Phys.*, 78:066001, 2015.

[183] E. Hallberg, E. Gagliano, and C. Balseiro. Finite-size study of a spin-1/2 heisenberg chain with competing interactions: Phase diagram and critical behavior. *Phys. Rev. B*, 41(13):9474–9479, 1990.

[184] T. Mishra, J. Carrasquilla, and M. Rigol. Phase diagram of the half-filled one-dimensional t-v-v' model. *Phys. Rev. B*, 84:115135, 2011.

[185] P. Weinberg and M. Bukov. Quspin: a python package for dynamics and exact diagonalisation of quantum many body systems part i: spin chains. *SciPost Phys.*, 2:003, 2017.

[186] P. Virtanen *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods*, 17:261–272, 2020.

[187] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum Natural Gradient. *arXiv:1909.02108 [quant-ph, stat]*, December 2019.

[188] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, February 2016.

[189] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):4213, September 2014.

[190] Edward Farhi and Hartmut Neven. Classification with Quantum Neural Networks on Near Term Processors. *arXiv:1802.06002 [quant-ph]*, August 2018.

[191] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, February 2020.

[192] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26, December 2019.

[193] Giuseppe Carleo and Matthias Troyer. Solving the Quantum Many-Body Problem with Artificial Neural Networks. *Science*, 355(6325):602–606, February 2017.

[194] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Recurrent Neural Network Wavefunctions. *arXiv:2002.02973 [cond-mat, physics:physics, physics:quant-ph]*, February 2020.

[195] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum Generative Adversarial Networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, December 2019.

[196] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, April 1982.

[197] Andrew J. Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D. Stevenson, and David J. Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 19(20):12585–12603, 2017.

[198] Levent Sagun, V. Ugur Guney, Gerard Ben Arous, and Yann LeCun. Explorations on high dimensional landscapes. *arXiv:1412.6615 [cs, stat]*, April 2015.

[199] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially No Barriers in Neural Network Energy Landscape. *arXiv:1803.00885 [cs, stat]*, February 2019.

[200] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. *arXiv:1706.04454 [cs]*, May 2018.

[201] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv:1609.04836 [cs, math]*, February 2017.

[202] Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the Hessian in deep neural networks. *arXiv:1902.02366 [cs, math, stat]*, February 2019.

[203] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, December 2018.

[204] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost-Function-Dependent Barren Plateaus in Shallow Quantum Neural Networks. *arXiv:2001.00550 [quant-ph]*, February 2020.

[205] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, December 2019.

[206] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[207] David Wierichs, Christian Gogolin, and Michael Kastoryano. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. *arXiv:2004.14666 [quant-ph]*, April 2020.

[208] 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.

[209] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On Optimization Methods for Deep Learning. page 8.

[210] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 598–605. 1990.

[211] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, page 164, 1993.

[212] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. *arXiv:1703.04730 [cs, stat]*, July 2017.

[213] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel

Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968 [physics, physics:quant-ph]*, February 2020.

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

[215] Patrick Huembeli and Alexandre Dauphin. PatrickHuembeli/vqc_loss_landscapes: ArXiv_version_v1.1. `https://github.com/PatrickHuembeli/vqc_loss_landscapes`, 2020.

[216] Kunal Sharma, M Cerezo, Lukasz Cincio, and Patrick J Coles. Trainability of dissipative perceptron-based quantum neural networks. *arXiv preprint arXiv:2005.12458*, 2020.

[217] Sukin Sim, Peter D. Johnson, and Alan Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, December 2019.

[218] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv:1609.04836 [cs, math]*, February 2017.

[219] R. C. M. Brekelmans, L. T. Driessen, H. J. M. Hamers, and D. den. Hertog. Gradient Estimation Schemes for Noisy Functions. *Journal of Optimization Theory and Applications*, 126(3):529–551, September 2005.

[220] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum Circuit Learning. *Physical Review A*, 98(3):032309, September 2018.

[221] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, March 2019.

[222] Kosuke Mitarai and Keisuke Fujii. Methodology for replacing indirect measurements with direct measurements. *Physical Review Research*, 1(1):013006, 2019.

[223] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond. *arXiv:1611.07476 [cs]*, October 2017.

[224] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *arXiv:1611.01838 [cs, stat]*, April 2017.

[225] Dawei Li, Tian Ding, and Ruoyu Sun. On the Benefit of Width for Neural Networks: Disappearance of Bad Basins. *arXiv:1812.11039 [cs, math, stat]*, January 2020.

[226] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The Break-Even Point on Optimization Trajectories of Deep Neural Networks. *arXiv:2002.09572 [cs, stat]*, February 2020.

[227] Anna Dawid, Patrick Huembeli, Michał Tomza, Maciej Lewenstein, and Alexandre Dauphin. Phase Detection with Neural Networks: Interpreting the Black Box. *arXiv:2004.04711 [cond-mat, physics:quant-ph]*, April 2020.

[228] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. Identifying quantum phase transitions with adversarial neural networks. *Phys. Rev. B*, 97:134109, Apr 2018.

[229] Jonas A. Kjäll, Jens H. Bardarson, and Frank Pollmann. Many-Body Localization in a Disordered Quantum Ising Chain. *Physical Review Letters*, 113(10):107204, September 2014.

[230] J. Goold, C. Gogolin, S. R. Clark, J. Eisert, A. Scardicchio, and A. Silva. Total correlations of the diagonal ensemble

herald the many-body localization transition. *Physical Review B*, 92(18):180202, November 2015.