Okinawa Institute of Science and Technology
Graduate University

Thesis submitted for the degree

Doctor of Philosophy

Machine Learning Applications for the Study and Control of Quantum Systems

by

Friederike Metz

Supervisor: **Professor Thomas Busch**

Jan 2023

Declaration of Original and Sole Authorship

I, Friederike Metz, declare that this thesis entitled *Machine Learning Applications for the Study and Control of Quantum Systems* and the data presented in it are original and my own work.

I confirm that:

- No part of this work has previously been submitted by me for a degree at this or any other university.
- References to the work of others have been clearly acknowledged. Quotations from the work of others have been clearly indicated, and attributed to them.
- In cases where others have contributed to part of this work, such contribution has been clearly acknowledged and distinguished from my own work.
- None of this work has been previously published elsewhere, with the exception of the following:
  - Chapter 2 has been published as [1]:

    Friederike Metz, Juan Polo, Natalya Weber, and Thomas Busch
    Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates
    Machine Learning: Science and Technology **2**, 035019 (2021)

    I implemented, trained, and evaluated the machine learning and wrote a first draft of the manuscript. All authors contributed to the discussions and to the editing of the manuscript draft.
  - Chapter 3 has been published as [2]:

    Korbinian Kottmann[*], **Friederike Metz**[*], Joana Fraxanet, Niccolò Baldelli
    Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer
    Phys. Rev. Research **3**, 043184 (2021)

[*]indicates co-first authorship

I implemented the quantum autoencoder in Qiskit (Python) and worked on the VQAD simulations for the TLFI and DEBHM model. All authors contributed to the discussions, the interpretation of the results, and the writing of the manuscript.

- Chapter 4 has been published as [3]:

  Aikaterini Gratsea, **Friederike Metz**, and Thomas Busch
  Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks
  Journal of Physics A: Mathematical and Theoretical **53**, 445306 (2020)

  I supervised this work and provided guidance, knowledge on reinforcement learning, and feedback. Furthermore, I contributed to the interpretation of the results, the writing of the manuscript, produced the final plots, and derived the asymptotic limit of the universally entangling coin sequence. All authors contributed to the discussions and the editing of the manuscript.

- Chapter 5 has been put as a preprint article on the arXiv [4]:

  Friederike Metz and Marin Bukov
  Self-Correcting Quantum Many-Body Control using Reinforcement Learning with Tensor Networks
  arXiv:2201.11790 [quant-ph] (2022)

  I performed the numerical simulations, the theoretical analysis, and wrote a first version of the manuscript. All authors contributed to the discussions, the interpretation of the results, and the editing of the manuscript draft.

- All of the above articles have been published in an open access format under the 'Creative Commons Attribution 4.0 International' license and I have permission to reprint them for the purpose of the thesis.

- During my PhD I also contributed to two different works, which are not part of this thesis [5, 6]. Their references are:

  Rashi Sachdeva[*], **Friederike Metz**[*], Manpreet Singh, Tapan Mishra, and Thomas Busch
  Two-leg-ladder Bose-Hubbard models with staggered fluxes
  Phys. Rev. A **98**, 063612 (2018)

  Karol Gietka, **Friederike Metz**, Tim Keller, Jing Li
  Adiabatic critical quantum metrology cannot reach the Heisenberg limit even when shortcuts to adiabaticity are applied
  Quantum **5**, 489 (2021)

Date: Jan 2023
Signature: [illegible]

*indicates co-first authorship

Abstract

Machine Learning Applications for the Study and Control of Quantum Systems

In this thesis, I consider the three main paradigms of machine learning – supervised, unsupervised, and reinforcement learning – and explore how each can be employed as a tool to study or control quantum systems. To this end, I adopt classical machine learning methods, but also illustrate how present-day quantum devices and concepts from condensed matter physics can be harnessed to adapt the machine learning models to the physical system being studied. In the first project, I use supervised learning techniques from classical object detection to locate quantum vortices in rotating Bose-Einstein condensates. The machine learning model achieves high accuracies even in the presence of noise, which makes it especially suitable for experimental settings. I then move on to the field of unsupervised learning and introduce a quantum anomaly detection framework based on parameterized quantum circuits to map out phase diagrams of quantum many-body systems. The proposed algorithm allows quantum systems to be directly analyzed on a quantum computer without any prior knowledge about its phases. Lastly, I consider two reinforcement learning applications for quantum control. In the first example, I use Q-learning to maximize the entanglement in discrete-time quantum walks. In the final study, I introduce a novel approach for controlling quantum many-body systems by leveraging matrix product states as a trainable machine learning ansatz for the reinforcement learning agent. This framework enables us to reach far larger system sizes than conventional neural network-based approaches.

Acknowledgment

First of all, I would like to thank my PhD supervisor Thomas Busch for all of his patience, advice, and guidance over the last years. Thanks for always having an open ear, cheering me up at the right times, enduring all my unconventional decisions, and for letting me freely explore my interests. This PhD has definitely been a lot easier with a caring and supportive advisor like you!

I was also very fortunate to have another brilliant supervisor and mentor in Marin Bukov. Thank you for always supporting, pushing and encouraging me, for including me in your group, and for all the valuable feedback and advice whether scientifically or for life. I am extremely grateful for all the time and effort you spent to make me a better physicist. I have learned a lot from you and hope to be able to pass it on one day.

Likewise, I also want to thank Juan Polo, Karol Gietka, Thomás Fogarty, and Rashi Sachdeva for being great supervisors, and for all their valuable insights, advice, and feedback.

Conversely, I also had the pleasure of helping in supervising two excellent students: Aikaterini Gratsea and Natalya Weber. I learned a lot from you in return and enjoyed our projects immensely. Thank you for that!

I want to thank all present and past members of the Quantum Systems Unit at OIST for being such a lighthearted and supportive group. In particular, I am grateful to Sahar, Tim, Hasan, Momo, Juan, Karol, Mossy, Chris, Houshu, Ayaka, James, Angela, Albert, Mathias, Lewis, Raul, Giedrius, and Eloisa for being there for me in fun but also difficult times. Special thanks go to Tim for all your help with my dog Moko, for enduring me in our office, and letting me have the desk with the window. I also want to thank Sawako for all the administrative support and for your assistance in many situations. Finally, I am grateful to James Schloss and Juan Polo for helping me with any programming related issues and to Jan Morén for providing the best HPC support.

During my PhD, I had the opportunity to stay a few months in Marin Bukov's group at the Max Planck Institute for the Physics of Complex Systems. I want to thank all people who made this stay such an enjoyable and rewarding experience, especially, Nicolò, Dominik, Luis, Paul, Benedikt, Pawel, and Joe. Also, I would like to thank my grandfather Volker and his wife Edith for making me feel at home in Dresden.

Similarly, I was also fortunate to do an internship at IBM in the course of this PhD. I thank all fellow intern friends and my team members in particular Aggie, Areeq, Pedro, Caleb, and Bryce for this unforgettable and enriching experience.

I want to thank my collaborators from ICFO, Korbinian Kottmann, Joana Fraxanet, and Niccolò Baldelli for a very productive and fun time and for simply being the best hackathon team members. Also, many thanks to Aikaterini, Korbinian, Patrick, Alexandre, Paolo, Borja, Niklas, Anna, and Marcin for including me in the QML reading group and for giving me such a warm welcome when I visited ICFO.

Outside of the academic life, I want to thank all of my friends in Okinawa and abroad who supported and cheered me up over the past 5 years. My biggest thanks go to Joel, Sandra, Leilee, Sahar, Tom, Rux, Sarah, Étienne, and Alessandro. I could not have done this without you!

I also thank my families in Germany and Japan for always believing in me and caring about me. I am especially grateful to my grandparents: Gisela and Friedhelm; the Neumann family: Petra and Klaus for being like parents to me; my brother Sebastian, my sister Annisa, my uncles, aunt and nephews: Frank, Ralf, Ulrike, Fabian, and Hendrik; my godmother Sabine and her husband Uwe. 浅野家の皆さん、お母さんの知子、お父さんの正、お姉さんの尚美、順子、紗世、そして家族の皆さん、本当にありがとうございました。いつも娘や妹のように気にかけてくれて、大変な時にそばにいてくれてありがとう。私の人生に皆さんがいてくれてとても幸運です.

Finally, I want to thank my mother Anke. Your strength, kindness, and your joy in life are an inspiration to me. Thank you for letting me pursue my dreams, for supporting me along the way, and for always being there for me.

Abbreviations

AE	Autoencoder
AMO	Atomic, molecular, and optical physics
AP	Average precision
BEC	Bose-Einstein condensate
CDW	Charge density wave
CNN	Convolutional neural network
DEBHM	Dimerized extended Bose Hubbard model
DMRG	Density-matrix renormalization group
DQN	Deep Q-Network
ES	Entanglement spectrum
GPE	Gross–Pitaevskii equation
MERA	Multi-scale entanglement renormalization ansatz
MI	Mott insulator
ML	Machine learning
MPO	Matrix product operator
MPS	Matrix product state
MSE	Mean-squared error
NISQ	Noisy intermediate-scale quantum
NN	Neural network
PCA	Principal component analysis
PEPS	Projected entangled pair states
QAE	Quantum autoencoder
QAOA	Quantum approximate optimization algorithm
QML	Quantum machine learning
RBM	Restricted Boltzmann machine
RL	Reinforcement learning
SPSA	Simultaneous perturbation stochastic approximation
SVD	Singular value decomposition
TEBD	Time-evolving block decimation
TLFI	Transverse longitudinal field Ising
TMI	Topological Mott insulator
TN	Tensor network
TTN	Tree tensor network
VQA	Variational quantum algorithm
VQAD	Variational quantum anomaly detection
VQE	Variational quantum eigensolver

To my mothers, Anke, Oma Gisela, Petra, and 知子
And to my fathers, Opa Friedhelm, Klaus, and 正

Contents

Declaration of Original and Sole Authorship **iii**

Abstract **v**

Acknowledgment **vii**

Abbreviations **ix**

Contents **xiii**

List of Figures **xvii**

List of Tables **xix**

Introduction **1**

1 Machine learning fundamentals **5**

1.1 Machine learning categories 5

1.1.1 Supervised learning 5

1.1.2 Unsupervised learning 6

1.1.3 Reinforcement learning 6

1.2 Classical machine learning 8

1.2.1 Neural networks 8

1.2.2 Convolutional neural networks 10

1.2.3 Autoencoders 12

1.3 Quantum-inspired machine learning 13

1.3.1 Tensor networks 14

1.3.2 Matrix product states 15

1.3.3 Tensor network-based machine learning 18

1.4 Quantum machine learning 20

1.4.1 Gate-based quantum computing 20

1.4.2 Quantum machine learning and NISQ 22

1.4.3 Variational quantum algorithms 24

1.4.4 Quantum autoencoder 27

1.5 Machine learning for quantum data 29

2 Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates 33
2.1 Introduction . . . 33
2.2 Physical system . . . 35
2.3 Machine learning model . . . 36
2.4 Training details . . . 38
2.4.1 Training data generation . . . 38
2.4.2 Neural network architecture and training . . . 40
2.4.3 Evaluation metrics . . . 41
2.4.4 Visualization of the CNN layers . . . 42
2.5 Results . . . 42
2.5.1 Vortex detection using density only . . . 42
2.5.2 Vortex detection using density and phase . . . 46
2.5.3 Generalization to different sources and levels of noise . . . 48
2.5.4 Generalization to different trap geometry . . . 49
2.6 Conclusion . . . 50

3 Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer 53
3.1 Introduction . . . 53
3.2 Proposal . . . 55
3.3 Results . . . 58
3.3.1 Simulations with ideal quantum data . . . 58
3.3.2 Experiments on a real quantum computer . . . 60
3.4 Training details . . . 63
3.5 Conclusion . . . 64

4 Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks 67
4.1 Introduction . . . 67
4.2 Background . . . 69
4.2.1 Quantum walk . . . 69
4.2.2 Q-learning . . . 71
4.3 Hybrid entanglement creation . . . 73
4.3.1 Universal entangling coin sequence . . . 73
4.3.2 Asymptotic limit of the universal entangling coin sequence . . . 76
4.3.3 Optimal coin sequences . . . 79
4.4 Conclusion . . . 82

5 Self-correcting quantum many-body control using reinforcement learning with tensor networks 85
5.1 Introduction . . . 85
5.2 Quantum many-body control . . . 88
5.3 Background . . . 88
5.3.1 Tabular Q-learning . . . 88
5.3.2 Deep Q-Learning . . . 90

5.4 Proposal . . . 91
5.4.1 Reinforcement learning (RL) framework . . . 91
5.4.2 Matrix product state ansatz for Q-learning (QMPS) . . . 91
5.5 Results . . . 93
5.5.1 Universal ground state preparation from arbitrary initial quantum states for $N = 4$ spins . . . 94
5.5.2 Preparation of a polarized product state from paramagnetic ground states for $N = 32$ spins . . . 95
5.5.3 Learning robust critical-region state preparation for $N = 16$ spins 98
5.6 NISQ device integration . . . 100
5.7 Additional details concerning the control studies . . . 105
5.7.1 Universal state preparation from arbitrary initial quantum states 107
5.7.2 Preparation of a polarized product state from paramagnetic ground states . . . 108
5.7.3 Learning robust critical-region state preparation . . . 113
5.8 Details of the QMPS architecture and training . . . 117
5.9 Conclusion . . . 120

Conclusion **123**

Bibliography **125**

List of Figures

1.1 Reinforcement learning feedback loop . . . 7
1.2 Perceptron and neural network model . . . 8
1.3 Convolutional neural network (CNN) . . . 11
1.4 Autoencoder (AE) . . . 13
1.5 Tensor network diagrams . . . 14
1.6 Matrix product state decomposition . . . 16
1.7 Order of contractions for computing MPS overlaps . . . 17
1.8 MPS ansatz for classification . . . 19
1.9 Exemplary quantum circuit . . . 22
1.10 The quantum-classical feedback loop of variational quantum algorithms 25
1.11 Quantum autoencoder . . . 28

2.1 Examples of BEC density and phase profiles in ground-state and non-equilibrium configurations . . . 36
2.2 CNN architecture for quantum vortex detection . . . 37
2.3 Distribution of the number of vortices in the training data . . . 39
2.4 Precision - recall curves . . . 42
2.5 Visualization of exemplary CNN feature maps . . . 43
2.6 Example vortex detection predictions on density images with and without noise . . . 45
2.7 Example vortex detection predictions on density and phase images with Gaussian random noise . . . 47
2.8 Example vortex detection predictions (including circulation directions) on density and phase images . . . 48
2.9 Example vortex detection predictions on density images of BECs in ring-shaped traps . . . 50

3.1 Overview of variational quantum anomaly detection (VQAD) . . . 55
3.2 Anomaly syndrome of VQAD trained on DEBHM ground states . . . 59
3.3 Anomaly syndrome of VQAD trained on DEBHM ground states in the presence of depolarizing noise . . . 60
3.4 Staggered magnetization and anomaly syndrome for the classically simulated TLFI model . . . 61
3.5 Staggered magnetization and anomaly syndrome computed on a quantum device for the TLFI model . . . 62

3.6 Comparison of the measurement outcomes with and without measurement error mitigation . . . 63

4.1 Diagrammatic representation of the 1D discrete time quantum walk. . . 70
4.2 Final Schmidt norms attained when evolving with the universal entangling coin sequence . . . 73
4.3 Asymptotic limit of the Schmidt norm under the universal entangling coin sequence . . . 74
4.4 Final Schmidt norms of the generalized universal entangling coin sequence 76
4.5 RL learning curves for generating hybrid entanglement . . . 80
4.6 Schmidt norm comparison of optimal and universal entangling coin sequences . . . 81
4.7 Final Schmidt norms achieved by RL optimized coin sequence . . . 82

5.1 QMPS framework for state preparation . . . 86
5.2 Many-body control studies in the ground state phase diagram of the quantum Ising model . . . 87
5.3 QMPS framework in detail . . . 92
5.4 Results of universal four-qubit control study . . . 95
5.5 Results of transverse-field Ising control study . . . 96
5.6 Results of mixed-field Ising control study . . . 99
5.7 MPS to circuit mapping . . . 101
5.8 QMPS framework as a hybrid quantum-classical algorithm . . . 102
5.9 State preparation success rates of the QMPS circuit framework for different number of measurement shots. . . . 103
5.10 State preparation success rates when testing the QMPS circuit framework on coherent and incoherent types of noise . . . 104
5.11 Results of universal single-particle control study . . . 107
5.12 Learning curves of the QMPS-1 agent for universal four-qubit control study . . . 108
5.13 Initial fidelities of transverse-field Ising control study . . . 109
5.14 Final achieved fidelities when reversing the state preparation scenario . 110
5.15 Protocols and corresponding fidelity and entropy curves when starting from different ground states . . . 110
5.16 Analysis of the optimal QMPS protocol obtained when starting from the initial ground state at $g_x = 1.01$ . . . 111
5.17 Obtained fidelities when testing QMPS protocols on different system sizes 112
5.18 SVD truncation error during training . . . 113
5.19 Initial fidelities of mixed-field Ising control study . . . 114
5.20 Half-chain von Neumann entanglement entropy of final states during training . . . 114
5.21 QMPS protocols and fidelity curves under noisy time evolution . . . 115
5.22 Fidelity curve comparison under noisy time evolution with fixed and adapted protocols . . . 117

List of Tables

2.1 CNN architecture 40
2.2 Vortex detector performance metrics 46
2.3 Vortex detector generalization performance metrics 49

5.1 RL environment parameters. 106
5.2 QMPS training hyperparameters. 118

Introduction

The field of machine learning has achieved remarkable successes in recent years ranging from agents beating the best human players at games [7], to deep neural networks that locate and classify objects in images with unprecedented accuracies [8], and to powerful language models that compose texts indistinguishable to those written by humans [9]. These impressive results have surged an interest in machine learning in a wide range of areas including the natural sciences and physics [10–12]. The first examples of machine learning techniques applied to condensed matter and quantum physics encompass supervised learning techniques to classify phases of matter [13, 14], unsupervised learning methods to variationally represent quantum many-body states [15], and reinforcement learning approaches to control quantum systems and protect them from noise [16, 17]. Since these pioneering works, machine learning methods have become a widely-used tool within the quantum domain. Modern areas of application include quantum phase detection [18–20], quantum state reconstruction [21], quantum error correction [22–24], quantum circuit optimization [25–28], and quantum control [16, 29–32].

In turn, ideas from statistical physics have been borrowed to develop an understanding and a theory of deep learning [10]. Additionally, tools from computational many-body physics and more specifically tensor networks, have been used as novel model architectures for machine learning tasks [33–35]. Tensor networks were originally developed for the efficient simulation of quantum many-body systems [36, 37]. These systems suffer from the so called *curse of dimensionality*, i.e., their Hilbert space dimension grows exponentially with the system size. Similarly, modern training data sets containing images or texts also represent high dimensional data and can potentially benefit from tensor network methods. Thus, tensor network-based machine learning techniques have been applied to typical learning tasks like classification [38–41] and generative modeling [42–44].

A completely different machine learning paradigm that recently emerged is quantum machine learning. Instead of utilizing parameterized neural networks that are evaluated on classical computers, quantum machine learning models are defined as parameterized quantum circuits on a quantum computer [45–48]. It is then expected, though still disputed, that quantum machine learning can lead to a speedup for certain tasks and/or leverage non-classical features to give rise to higher accuracies than attainable with fully classical techniques. At present, quantum machine learning is still in its infancy mostly since current quantum devices are still too noisy and small-scale to fully explore and understand their capabilities [49]. Despite these challenges, quantum machine learning techniques have already been applied to various supervised and unsupervised learning tasks [47, 50–52].

Machine learning presents a promising tool to advance science and to help us in solving various open problems in physics and the quantum technologies in particular. Whether it is to automate or optimize certain tasks, detect patterns in data, or discover new physics, machine learning has numerous potential areas of application. In this thesis, I will discuss four such examples of machine learning applications that assist us in studying and controlling quantum systems. I will cover applications from all three areas of machine learning, i.e., supervised, unsupervised, and reinforcement learning. Furthermore, I will demonstrate how classical machine learning techniques, quantum-inspired ones, and fully quantum machine learning approaches can each be effectively harnessed when applied to quantum mechanical systems.

In the first project, I devise a deep learning-based object detection framework that is able to locate quantum vortices in density snapshots of atomic Bose-Einstein condensates (BECs) [1]. Similar to their classical counterpart, the dynamics of quantum vortices gives rise to rich and complex non-equilibrium processes like turbulence and chaos [53–59]. However, as a prerequisite for the study of these out-of-equilibrium phenomena we require to trace the precise positions of all vortices in the BEC over time [60]. This task is significantly more challenging for experimental data, where often only the density profile of the wavefunction is available, and effects of noise and finite temperatures blur the characteristic features of vortices. Hence, I propose a supervised machine learning model that is motivated by state-of-the-art computer vision techniques for predicting the precise positions of all vortices in simulated BEC density images. Our framework achieves high test set accuracies and is able to locate vortices both in ground state and in the more challenging out-of-equilibrium configurations. Furthermore, the detection is robust to different sources of noise that commonly arise in experimental settings.

In the next study, I turn to an unsupervised learning application and introduce a quantum anomaly detection framework that allows us to map out phase diagrams of quantum many-body systems directly on quantum hardware [2]. Classifying phases of matter and predicting the location of their phase boundaries often requires expert human knowledge and prior intuition about the physical system at hand, such as which order parameters to analyze [61]. It would hence be desirable if this task can be automated and potentially lead to the discovery of novel phases of matter [18, 62–65]. Furthermore, with the possibility of performing large-scale quantum simulations on near-term quantum computers grows the need for quantum algorithms that let us investigate these quantum systems natively on the physical devices [49]. In this work, I and my collaborators propose a quantum analog of anomaly detection that is based on a quantum autoenoder. The variational quantum algorithm is able to discover the phases of the paradigmatic transverse-longitudinal field Ising model [66, 67] and the extended Bose Hubbard model with dimerized hoppings [68]. Moreover, we show that the framework can already be leveraged on present-day quantum computers by performing it on one of the IBM Quantum devices.

In the last two projects, I showcase how reinforcement learning can be employed for solving quantum control problems. In the first project, I use tabular Q-learning to maximize the entanglement generation in discrete-time quantum walks [3]. Entanglement can be regarded as a resource for many quantum information processing tasks such as quantum computation, quantum communication, or quantum teleportation [69]. Each of these applications have a physical realization in terms of the discrete-time quantum walk [70–75]. Hence, it is important to find fast and robust schemes for creating highly entangled states such that quantum walks can be leveraged in optimal ways. To that end, I and my co-authors devise a reinforcement learning agent that learns to prepare highly entangled states over a class of different initial states. Moreover, we introduce a deterministic sequence of quantum walk coin operators which can universally generate high entanglement irrespective of a localized initial state.

In the last study, I apply reinforcement learning techniques to the control of quantum many-body systems [4]. As previously mentioned, these systems suffer from the curse of dimensionality which makes it infeasible to exactly simulate large systems on classical computers. Consequently, it is even more challenging to devise optimal control strategies thereof. Quantum many-body control, however, is essential for quantum technologies that are nowadays based on harnessing a large number of correlated particles [76–80]. Hence, in this work I propose a novel Q-learning framework in which both the quantum states as well as the agent are represented by tensor networks. This learning architecture allows us to efficiently train the quantum state-aware agent on large system sizes which would be inaccessible with conventional neural network approaches. Furthermore, I show that the trained agents can generalize their protocols to unseen states, are robust to noise, and can self-correct protocols as the system is being time-evolved.

This thesis is structured around the four main projects each of which forms the basis of one chapter. A brief outline of the thesis is given below.

- **Chapter 1** provides the fundamentals of machine learning techniques that form a prerequisite for the subsequent chapters. Additionally, some of the past successes of machine learning applications in the quantum domain are discussed.

- **Chapter 2** is based on the content of publication [1]: Friederike Metz, Juan Polo, Natalya Weber, and Thomas Busch, *Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates*, Machine Learning: Science and Technology **2**, 035019 (2021).

- **Chapter 3** is based on the content of publication [2]: Korbinian Kottmann, Friederike Metz, Joana Fraxanet, and Niccolò Baldelli, *Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer*, Phys. Rev. Research **3**, 043184 (2021).

- **Chapter 4** is based on the content of publication [3]: Aikaterini Gratsea, Friederike Metz, and Thomas Busch, *Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks*, Journal of Physics A: Mathematical and Theoretical **53**, 445306 (2020).

- **Chapter 5** is based on the content of the preprint article [4]: Friederike Metz and Marin Bukov, *Self-Correcting Quantum Many-Body Control using Reinforcement Learning with Tensor Networks*, arXiv:2201.11790 [quant-ph] (2022).

Chapter 1

Machine learning fundamentals

Machine learning is a subfield of artificial intelligence and can be defined as learning from data or experience. The modern form of machine learning does not require features or rules to be explicitly programmed. Rather, it is often synonymous with a model that is trained, i.e, optimized over some data examples. The optimized model then provides the solution to the problem by evaluating it on different data inputs. In this chapter, I will introduce the basics of (quantum) machine learning. I will focus on only those concepts that are relevant for the main part of this thesis. For a comprehensive introduction to machine learning and its applications in the physical sciences I refer to Refs. [10–12, 81–83].

I will start by revisiting the three main categories of machine learning in Section 1.1. Section 1.2 introduces classical machine learning models, i.e., neural networks and its variants. Section 1.3 establishes tensor networks as quantum-inspired machine learning ansatze. In Section 1.4 I will discuss models that are evaluated on quantum devices which are therefore referred to as quantum machine learning models. Finally, in Section 1.5 I will review some of the most famous applications of machine learning in the quantum sciences.

1.1 Machine learning categories

The current zoo of machine learning algorithms can be loosely separated into three main paradigms according to the structure of the available training data. These are *supervised*, *unsupervised*, and *reinforcement learning*. Note that in recent years other categories have emerged as well, such as semi-supervised and self-supervised learning.

1.1.1 Supervised learning

Supervised learning describes the branch of machine learning where one has access to labeled training data $\mathcal{D} = \{(\mathbf{x}_i, \hat{\mathbf{y}}_i)\}$, i.e., for every training data input $\mathbf{x}_i$, a correct label or output $\hat{\mathbf{y}}_i$ is known. The task is to learn a model that given different inputs $\mathbf{x}$ predicts the correct outputs $\hat{\mathbf{y}}$. The labels can be continuous numbers in which case the task is referred to as regression, or discrete numbers which corresponds to a classification problem. For example, predicting house prizes from features of the house like size, number of rooms, etc. would be considered a regression problem while labeling objects contained in images is a classification task. Note that there are also situations which involve both regression and classification, e.g., imagine we not just want to label an object in an image, but also precisely locate it within said image. This scenario is commonly known as object detection.

The models that are being learned to predict labels from inputs are usually comprised of parameterized functions $f_\theta(\mathbf{x})$ which can range from simple linear models with just a handful of parameters θ to highly complex deep neural networks with billions of parameters. The learning or training of a machine learning model then amounts to optimizing the parameters θ such that the mapping between the data input $\mathbf{x}$ and output $\mathbf{y} = f_\theta(\mathbf{x})$ is most accurate. The performance of a model is often measured in terms of a cost or loss function C. In the case of regression a widely used cost function is the mean squared error (MSE) between the predicted labels $\mathbf{y}$ and the ground truth $\hat{\mathbf{y}}$

$$C = \frac{1}{N_T} \sum_{i=1}^{N_T} |\mathbf{y}(\mathbf{x}_i) - \hat{\mathbf{y}}_i|^2, \tag{1.1}$$

where N_T denotes the size of the training data set. However, other cost functions like the cross entropy might be more suited for a given task. Minimizing a loss function is the central objective in most machine learning algorithms including the cases of unsupervised and reinforcement learning.

1.1.2 Unsupervised learning

Unsupervised learning applies to those data sets $\mathcal{D} = \{(\mathbf{x}_i)\}$ that do not contain any labels, i.e., no extra information about the data is supplied. This is the default situation for most collected data since labeling usually requires human experts, is time-consuming, and often not possible at all. The goal of unsupervised learning is therefore, loosely speaking, to learn some patterns about the data at hand. Hence, the tasks are diverse and include a range of concepts such as dimensional reduction, clustering, density estimation, generative modeling, anomaly detection etc. For example, in density estimation the probability distribution underlying the data is learned, generative modeling allows new, similar data to be generated and anomaly detection finds outliers in otherwise homogeneous data sets. Especially, the field of generative modeling has gotten widespread attention due to the generation of artificial photos, art, and text that are indistinguishable from human created ones [9, 84, 85].

1.1.3 Reinforcement learning

The last of the three major ML categories – reinforcement learning – deals with the problem of control tasks and games [86]. No data is a priori required, but rather the data or experience is collected as part of the algorithm. Learning is enabled via interactions and feedback in a trial and error fashion in analogy to how dogs are taught to sit.

Most RL problems are phrased in the language of an agent and environment (see Fig. 1.1). The agent is the abstract entity making decisions and performing actions

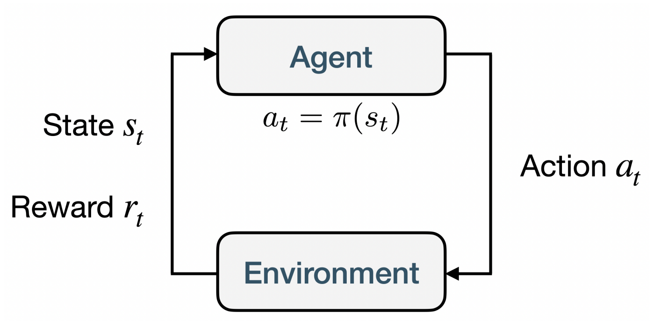


Figure 1.1: Reinforcement learning feedback loop between agent and environment. At each time step t the agent receives information about the current state s_t of the environment. Depending on this state, the agent then chooses an action a_t via a policy function π. In turn, the environment transitions to a new state s_{t+1} depending on the previous state s_t and performed action a_t. Additionally, a reward signal r_t is computed at each time step and supplied to the agent to improve the policy π.

while the environment is the system that is being controlled. The interactions between agent and environment occur in discrete time steps. At each step t the agent can observe the current state s_t of the environment. Depending on this state, the agent then chooses to perform a specific action a_t. The function that maps states to actions is commonly referred to as policy π and can be understood as the strategy that the agent follows. Each applied action in turn changes the state of the environment. Despite the updated state information, the agent also receives a scalar reward signal r_t at each time step that guides the agent to perform improved actions, that is, the reward reinforces "good" behavior. The reward can also be negative in which case it serves as a punishment. The resultant feedback loop between agent and environment constitutes a Markov Decision Process (MDP).

The overall goal in RL is to maximize the expected sum of future rewards by finding an optimal policy π^*, i.e., by always acting optimally in each state. There are various RL algorithms that can be leveraged for this task. Model-based RL algorithm use or build a model of the environment as part of the training while in model-free RL techniques the environment is considered a black-box. Furthermore, there are policy-gradient approaches where the optimal policy is explicitly learned and value-function-based approaches where the optimal policy is found implicitly by learning so called Q-values. The choice of RL algorithm depends on various factors, such as whether the state and action spaces are discrete or continuous or how time-expensive interactions with the environment are.

In modern RL tasks the policy (or equivalently the Q-value) are often approximated by parameterized functions which are then optimized as part of the algorithm. Similarly to how neural networks have revolutionized the fields of (un)-supervised learning, deep RL has led to some of the most impressive results in control tasks and game playing. Famous examples of the latter are the success of deep RL agents in Atari games, Starcraft, and the board-game Go [7, 87, 88].

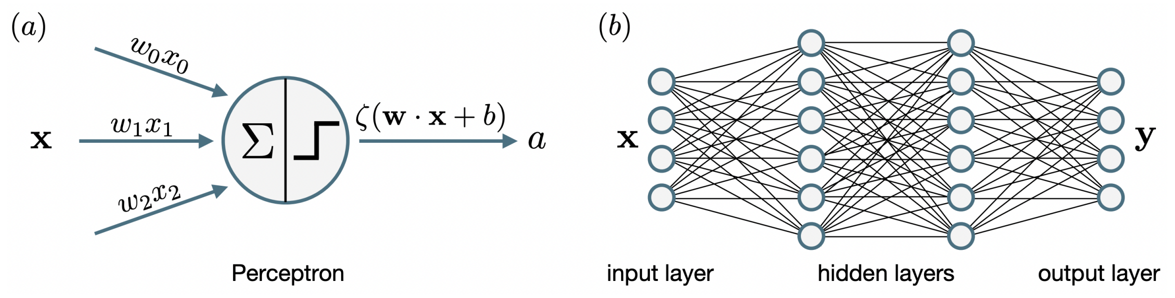


Figure 1.2: (a) Sketch of a single artificial neuron (perceptron). It computes the dot product of the input data $\mathbf{x}$ and the trainable weights $\mathbf{w}$, adds a bias, and applies a nonlinearity ζ. The output is often called the activation a of the neuron. (b) Exemplary sketch of an all-to-all connected neural network with two hidden layers. Each edge corresponds to a single neuron, while vertices represent the weight matrices.

1.2 Classical machine learning

The first class of machine learning (ML) models I introduce are fully classical ones, i.e., standard and widely-used models within the machine learning community. These are stored, evaluated, and trained on classical computers (e.g. on a CPU or GPU). ML models can be divided into two categories: deep learning models and those not based on neural networks. Examples of the latter are a simple linear model, support vector machines or decision trees. In the following, I will focus on neural network-based approaches and introduce the basics of neural networks, convolutional neural networks (CNN), and autoencoders (AE). A more comprehensive introduction to the field of deep learning can be found in Refs. [89–93].

1.2.1 Neural networks

Artificial neural networks are non-linear, parameterized functions inspired by the neural networks in our brains. An NN processes the input data sequentially via several layer-wise transformations. The basic building block of an NN is a single neuron (or perceptron) as shown in Fig. 1.2(a). It takes a vector $\mathbf{x}$ as input, computes its inner product with a weight vector $\mathbf{w}$, adds a scalar bias b, and feeds the result through a nonlinear function ζ

$$a = \zeta\left(\sum_i w_i x_i + b\right). \tag{1.2}$$

The scalar output signal a is called the activation of the neuron. A neural network layer is composed of many such neurons so that we can express the combined activation output vector $\mathbf{a}$ over all neurons after each layer as $\mathbf{a} = \zeta(W\mathbf{x} + \mathbf{b})$, where W and $\mathbf{b}$ have been promoted to a matrix and vector respectively. For convenience, I define $z_i \equiv \sum_j w_{ij}x_j + b_i$ to be the affine part of the perceptron transformation.

The first layer of a neural network takes the data $\mathbf{x}$ as input and is therefore called input layer while all subsequent layers l take the activations $\mathbf{a}^{l-1}$ from the previous layer $l-1$ as their input. The activations $\mathbf{a}^L$ of the final output layer L provide the model output (see Fig. 1.2(b)). The intermediate layers are commonly referred to as hidden layers and the number of hidden neurons in each layer is called their hidden dimension. Note that the number of neurons in the input and output layers are determined by the problem, i.e, the dimensions of the input and output spaces. The weights and biases of all layers together comprise the trainable parameters of the neural network.

The non-linear transformation ζ after each layer operation is essential and, in fact, gives deep NNs much of their expressive power. Without any non-linearities the NN computation would reduce to a simple affine transformation and would therefore be able to represent only very limited dependencies. However, NNs with a single non-linear hidden layer can already serve as universal function approximators [94, 95]. In practice, deep NNs with many hidden layers but small hidden dimensions turn out to be more expressive than shallow neural networks with the same number of parameters, i.e, a large hidden dimension. This observation gave rise to the field of *deep* learning with architectures comprised of many hidden layers. Common choices for the non-linear activation function ζ are the rectified linear unit (ReLU) $\zeta(z) = \max(0, z)$, and the tanh function $\zeta(z) = \tanh(z)$. Note that the nonlinearity is conventionally applied elementwise.

The parameters in a neural network are usually optimized via gradient descent with the goal of minimizing some cost function C. Gradient descent is an iterative method for finding local minima of an objective function. In each step the model parameters are updated by moving in the opposite direction of the gradient of the cost function

$$\theta \leftarrow \theta - \alpha \frac{\partial C(\theta)}{\partial \theta}, \tag{1.3}$$

where α represents the learning rate which is typically chosen small enough to prevent overshooting the local optima.

The gradients of the cost function with respect to each of the parameters are obtained through backpropagation which is an efficient, reverse automatic differentiation technique for computing derivatives of nested functions [96]. In its essence, backpropagation amounts to repeatedly applying the chain rule. For backpropagation to be applicable we need to assume that the cost function can be written in terms of the NN output $\mathbf{y}(\mathbf{x})$ and is represented as a sum (or average) over different training examples, i.e.,

$$C = \frac{1}{N_T} \sum_{n=1}^{N_T} g\left(\mathbf{y}(\mathbf{x}_n), \hat{\mathbf{y}}_n\right), \tag{1.4}$$

where $\hat{\mathbf{y}}_n$ is the ground-truth label for a given input $\mathbf{x}_n$. This assumption is valid for most cost functions including the mean-squared error (MSE) $g = |\mathbf{y}(\mathbf{x}_n) - \hat{\mathbf{y}}_n|^2$. The partial derivative of the cost function with respect to a particular parameter θ (weight or bias) therefore computes to

$$\frac{\partial C}{\partial \theta} = \frac{1}{N_T} \sum_{n=1}^{N_T} \sum_i \frac{\partial g}{\partial y_i(\mathbf{x}_n)} \frac{\partial y_i(\mathbf{x}_n)}{\partial \theta} . \tag{1.5}$$

The backpropagation algorithm traverses the NN in reverse and first computes the gradients with respect to the weights and biases in the last layer L

$$\frac{\partial y_i}{\partial b_j^L} = \frac{\partial \zeta}{\partial z_j^L}\delta_{ij}, \tag{1.6}$$

$$\frac{\partial y_i}{\partial w_{jk}^L} = \frac{\partial \zeta}{\partial z_j^L}\delta_{ij} a_k^{L-1}, \tag{1.7}$$

where δ_{ij} is the Kronecker delta and the dependence on the input data $\mathbf{x}_n$ has been omitted. Similarly, we can derive an expression for the derivatives for all subsequent layers

$$\frac{\partial y_i}{\partial b_j^l} = \gamma_{ij}^l, \tag{1.8}$$

$$\frac{\partial y_i}{\partial w_{jk}^l} = \gamma_{ij}^l a_k^{l-1}, \tag{1.9}$$

where

$$\gamma_{ij}^l = \sum_k \gamma_{ik}^{l+1} w_{kj}^{l+1} \frac{\partial \zeta}{\partial z_j^l}, \qquad \text{and} \qquad \gamma_{ij}^L = \frac{\partial \zeta}{\partial z_i^L}\delta_{ij}. \tag{1.10}$$

The last two equations represent the recursive application of the chain rule. Note that to compute all gradients we require the activations $\mathbf{a}^l$ to be known for each input data point $\mathbf{x}_n$ and each layer l. Hence, we first need to evaluate the NN on the input data which is called a forward pass. Accordingly, the gradient taking step is referred to as the backward pass. Optimizing all parameters in a conventional neural network requires only one forward and one backward pass both of which are efficient in that they have run times scaling only linear in the number of parameters [97]. Due to this reason, backpropagation together with modern, hardware-accelerated automatic differentiation libraries opened the door to the use of deep learning architectures with billions of parameters.

1.2.2 Convolutional neural networks

The structure of the standard, fully-connected neural network introduced above does not take into account any properties that the input data might have. However, for some tasks and forms of data we already know that certain structures are present and can be exploited. Consider for example the problem of finding dogs in images. The features that define a dog (e.g. tail, snout, legs) take up only certain parts of an image and hence are inherently local. Moreover, the precise position of the dog within an image should not matter as the neural network makes its prediction. Thus, we can assume that the model should be translationally invariant. A type of neural network that explicitly leverages locality and translational invariance are convolutional neural networks (CNN) which are therefore particularly suited for image processing tasks [90, 98, 99].

For simplicity, I will introduce CNNs for the case of two dimensional (2d) input

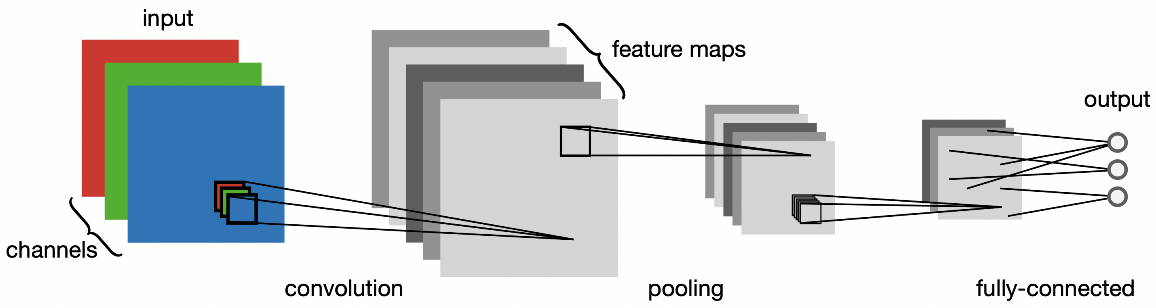


Figure 1.3: Visualization of a convolutional neural network (CNN) with two convolutional, one pooling, and one final fully-connected layer. In a convolutional layer a kernel or filter is applied to small patches of the input images (along all channels) which is then strided along the full image to produce a feature map. Different kernels give rise to distinct features each represented by another channel. The pooling operation reduces patches of the input image to a single number via taking their mean or max values.

data, i.e., images. However, CNNs can straightforwardly be generalized to one or higher dimensions. A CNN consists of several layers each of which can perform one of two operations: a convolutional or a pooling transformation (see Fig. 1.3). Importantly, each of these layers maps a 2d input image back onto a two dimensional space called feature map. Similarly to the standard fully-connected NN, in a convolutional layer we apply a weight matrix $w^{j_x,j_y}_{i_x,i_y}$ that maps the pixel value p_{i_x,i_y} at positions i_x, i_y of the input image to a pixel value p_{j_x,j_y} at positions j_x, j_y of the output image: $z^{j_x,j_y} = \sum_{i_x,i_y} w^{j_x,j_y}_{i_x,i_y} p_{i_x,i_y} + b^{j_x,j_y}$. However, we restrict the weight matrix to be translationally invariant by allowing it to only depend on pixel position differences rather than absolute positions, i.e., $w^{j_x,j_y}_{i_x,i_y} = w(i_x - j_x, i_y - j_y)$. Hence, the resultant weight matrix elements automatically share some of their values. On top of that, we also set those weight matrix elements to zero for which $|i_x - j_x| > K, |i_y - j_y| > K$, where K is called the kernel size[*]. The latter restriction makes the model local as an output pixel only depends on a small local patch of input pixels.

Instead of thinking in terms of weight matrices, we can also adopt the notion of kernels (or equivalently filters) as is common in computer vision. The truncated weight matrix $w^{0,0}_{i_x,i_y}$ defines the kernel and to obtain each output pixel value p_{j_x,j_y} we simply stride it over the input image. Besides the kernel size K, we also have to specify the stride S which determines how many pixels are skipped when the kernel is moved over the input image. Both of these hyperparameters can have different values for each layer and have to be determined for each problem separately. Finally, each pixel in a two dimensional image is usually not defined by one value, but rather by its three red, green, blue (RGB) color channels. Similarly, one is usually interested in extracting several distinct feature maps from the input and hence a convolutional layer outputs a number of different channels C. Therefore, the kernel at a given layer l is in fact a 4d tensor of dimensions $K_l \times K_l \times C_{l-1} \times C_l$.

[*]In principle, the kernel size K can be different in the x and y directions. However, in practice it is often chosen to be symmetric.

The other operation that is typically present in CNNs is pooling. Here, the input image is coarse-grained, i.e., its resolution is reduced, similarly to the block decimation procedure in the real-space renormalization group. We divide the input image into small equally sized patches and reduce each patch to a single number in the output map by, for example, choosing the maximum value, or taking the mean over all pixels in a patch. In contrast to convolutional layers, the pooling operation is conventionally applied separately to each input channel and thus does not change the number of channels. Also note that pooling does not involve any trainable parameters. Pooling layers are usually inserted only at a few locations in the CNN. The convolutional and pooling operations within a CNN are commonly followed by a final fully-connected layer before the output can be read off (see Fig. 1.3).

The advantage of CNNs does not only lie in their ability to leverage short-range correlations, local features, and symmetries in the input data, but also that the inherent weight sharing mechanism greatly reduces the overall number of optimizable parameters. Hence, deep CNNs can be trained much more efficiently than their dense, fully-connected NN counterpart while reaching the same or even better test set accuracies. CNN architectures have been used to produce some of the most impressive results within deep learning. A famous example is image recognition where images of a large class of objects have to be annotated (e.g. using the ImageNet data set) [100]. Another example is the more challenging problem of object detection where several objects can be present within an image and the task is to locate and annotate each of them [8].

1.2.3 Autoencoders

Neural networks and its variants like the CNN can be employed as parameterized ansatze for any kind of machine learning task. However, there also exist special networks that are designed for specific use cases. The autoencoder (AE) is one such example [90, 101, 102]. It was developed for unsupervised learning, i.e, situations where no labels for the training data are present, and can solve a variety of different tasks like representation learning, dimensional reduction, denoising, or anomaly detection.

An autoencoder consists of two parts, an encoder and a decoder which can be represented by standard neural networks, CNNs, or other types of networks. The encoder f_θ takes the d-dimensional data vector $\mathbf{x}$ as input and maps it onto an m-dimensional space, called latent space: $\mathbf{z} = f_\theta(\mathbf{x})$. The decoder g_ϕ in turn takes the latent space representation $\mathbf{z}$ and maps it back to the original space: $\hat{\mathbf{x}} = g_\phi(\mathbf{z})$. The goal of the AE is to reconstruct the original input data at the output $\mathbf{x} \simeq \hat{\mathbf{x}}$ (see Fig. 1.4). This can be achieved by finding optimal encoder and decoder parameters θ and ϕ that minimize the mean-squared error between model in-and outputs

$$C = \frac{1}{N_T} \sum_{n=1}^{N_T} |\mathbf{x}_n - \hat{\mathbf{x}}_n|^2. \tag{1.11}$$

In principle, it is trivial to find a function that exactly reproduces the input for all training data points; in fact the identity operation would always satisfy the requirement by definition. However, the autoencoder has the special property that the latent space dimension is chosen much smaller than the input dimension $m \ll d$ (see Fig. 1.4).

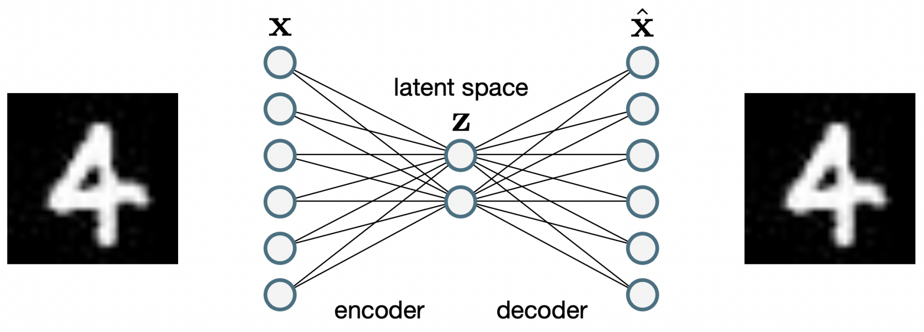


Figure 1.4: The autoencoder takes an input $\mathbf{x}$, encodes it into a latent space variable $\mathbf{z}$ on a low-dimensional manifold, and subsequently decodes the latent space representation again to reproduce the original input $\mathbf{x} \simeq \hat{\mathbf{x}}$. The latent space serves as a bottleneck through which information about the input has to pass.

Hence, the information about the input has to pass through a low dimensional manifold, a bottleneck, before it can be reconstructed again. The encoder therefore has to learn to imprint the most characteristic features of the data onto the latent space such that the maximum amount of information can be used by the decoder to accurately reproduce the input.

After successful training, the encoder of the AE can be used as a feature extractor or dimensional reduction technique for data that is similar to the training data [102]. AEs can also be leveraged for the task of anomaly detection [103]. In that case, we train the autoencoder on a homogeneous training data set for which we minimize the reconstruction error [Eq. (1.11)]. When testing the AE afterwards on anomalous data that possesses different features than the training data, the AE will fail in reconstructing the inputs and give rise to a large reconstruction error. A peak in the reconstruction loss can therefore be regarded as a syndrome of data anomalies.

Autoencoders have also been used for denoising images [104]. In this context, the AE takes as input a noisy image and has to output the corresponding denoised one. To that end, it is trained by minimizing the error between the original noise-free image and the model output. Similar methods can for example also be applied to the task of coloring black-and-white images [105]. Finally, with a few modifications autoencoders, or specifically variational autoencoders, can be leveraged for generative modeling, i.e, for generating completely new data examples that are similar to the original training data [106].

1.3 Quantum-inspired machine learning

This section is devoted to yet another class of classical machine learning models which are, however, inspired by methods developed within the field of computational quantum many-body physics. The main motivation stems from the fact that both, machine learning and quantum many-body physics deal with high-dimensional data that has to be processed in efficient ways. In the following, I will introduce the basics of tensor networks and the matrix product state as a specific example for a quantum-inspired

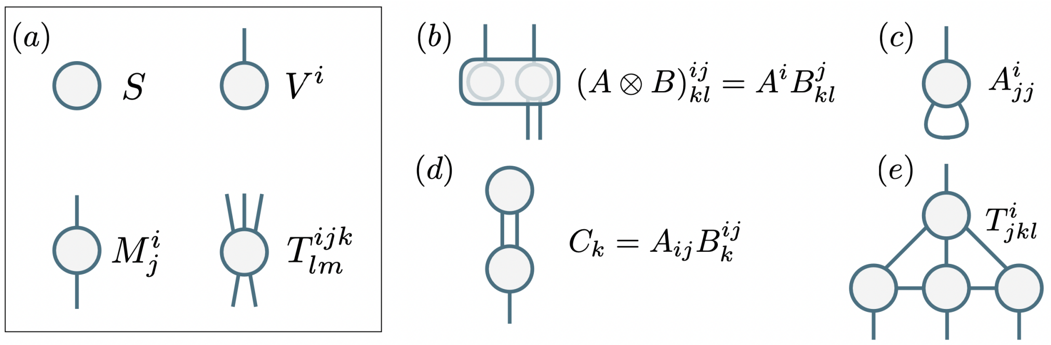


Figure 1.5: Pictorial representation of tensor networks and tensor operations. (a) Visualization of a scalar (rank-0 tensor), vector (rank-1 tensor), matrix (rank-2 tensor) and a rank-5 tensor. Common tensor operations are the (b) tensor product, (c) the trace, and (d) the contraction of two or more tensors. (e) Example of a tensor network that results in a rank-4 tensor after contracting over all closed indices.

machine learning ansatz.

1.3.1 Tensor networks

Tensors can be considered as a generalization of vectors and matrices. A tensor $T_{i_1,\dots,i_r}$ with r indices of dimensions $d_1 \times \cdots \times d_r$ is defined as an element of $\mathbb{C}^{d_1 \times \cdots \times d_r}$. The number of indices of a tensor is also commonly called its rank. A convenient way of visualizing tensors and algebraic operations thereof are tensor diagrams as shown in Fig. 1.5 [37, 107–110]. Using this notation, a tensor is usually denoted by a vertex (e.g. a circle, square, or triangle) and a number of legs equal to the rank of the tensor. Fig. 1.5(a) shows the diagram for a scalar, vector, matrix, and rank-5 tensor respectively.

All common tensor operations can be expressed using the diagrammatic approach. The tensor product defined by

$$[A \otimes B]_{i_1,\dots,i_r,j_1,\dots,j_s} := A_{i_1,\dots,i_r} \cdot B_{j_1,\dots,j_s} \tag{1.12}$$

corresponds to the tensors being put next to each other (Fig. 1.5(b)). The trace defined by summing over two common indices[*]

$$[\mathrm{Tr}_{a,b}\, A]_{i_1,\dots,i_{a-1},i_{a+1},\dots,i_{b-1},i_{b+1},\dots,i_r} = A_{i_1,\dots,i_{a-1},\alpha,i_{a+1},\dots,i_{b-1},\alpha,i_{b+1},\dots,i_r} \tag{1.13}$$

is represented by the corresponding legs of the tensor being joined and forming a loop (Fig. 1.5(c)). And similarly, the contraction of two tensors, e.g.,

$$C^{j_1,\dots,j_s}_{i_1,\dots i_r} = A_{i_1,\dots\alpha\dots i_r} B^{j_1,\dots,\alpha,\dots,j_s} \tag{1.14}$$

is denoted by the respective legs being joined (Fig. 1.5(d)). Tensor contractions include the vector-dot product, matrix-vector multiplication, and matrix-matrix multiplication

[*]We use the Einstein convention, i.e, repeated indices are summed over.

as special cases.

We are always able to change the rank of a tensor by reshaping the indices due to the fact that vector spaces are isomorphic as long as the total dimensions are equal. In the diagrammatic representation, reshaping a tensor corresponds to either the grouping or splitting of legs depending on whether the rank is lowered or raised. Note that while the positioning of the legs in a tensor diagram is usually arbitrary, we will follow a convention where co-and contra-variant indices point in opposite directions. This ensures that only valid contractions between for example a bra and ket vector are possible.

A tensor network refers to the collection of several tensors (see Fig. 1.5(e) for an example). Open legs of the network determine the indices of the composite tensor and therefore its rank. Closed legs, i.e., legs that start and end in a tensor, are contracted over. The value of the resultant combined tensor can be computed by performing the corresponding index summations as indicated by the network connections. While the order in which indices are contracted does not affect the final result, it usually plays a significant role in terms of the involved computational complexity. In fact, finding the optimal order of contractions for an arbitrary network is shown to be NP-complete [111, 112]. Examples of tensor networks allowing for efficient computations are matrix product states (MPS) [113–116], tree tensor networks (TTN) [117], and the multi-scale entanglement renormalization ansatz (MERA) [118, 119].

1.3.2 Matrix product states

Matrix product states (MPS) were originally developed as a numerical tool for the efficient simulation of one dimensional quantum many-body systems [113–116, 120]. In parallel, they have also been known within the applied mathematics community as tensor trains [121]. In the following, I will briefly introduce MPS from a physicist's viewpoint. For a more comprehensive review I will refer to Refs. [36, 37, 107, 108, 122].

Consider a quantum system that is defined on a one dimensional lattice, such as a linear chain of N spins with a local Hilbert space dimension d_i at site i ($d_i = 2$ in the case of spin-1/2). The quantum state $|\psi\rangle$ of the full Hilbert space $H = \mathbb{C}^{d_1 \times \ldots \times d_N}$ can be represented as a rank-N tensor

$$|\psi\rangle = \sum_{j_1,\ldots,j_N} \psi_{j_1,j_2,\ldots,j_N} |j_1, j_2, \ldots, j_N\rangle \,, \tag{1.15}$$

where $j_i \in \{1, \ldots, d_i\}$. Note that the number of parameters in this ansatz and consequently the memory resources scale exponentially with the system, i.e., for a uniform local Hilbert space dimension $d_i = d$, we deal with a d^N dimensional vector – this is commonly referred to as the curse of dimensionality.

A matrix product state is a decomposition of the rank-N tensor into a collection of N rank-3 tensors as depicted in Fig. 1.6. In the first step of the decomposition, we group all but the first index together to obtain a matrix $\psi_{j_1,k}$ and apply a singular value decomposition (or equivalently a Schmidt decomposition) to it: $\psi_{j_1,k} = U_{j_1,\alpha} S_\alpha V^*_{\alpha,k}$, where U and V are isometries. Next, we take the tensor V, group the indices (α, j_2) and $(j_3, \ldots, j_N)$ together and perform another singular value decomposition (SVD)

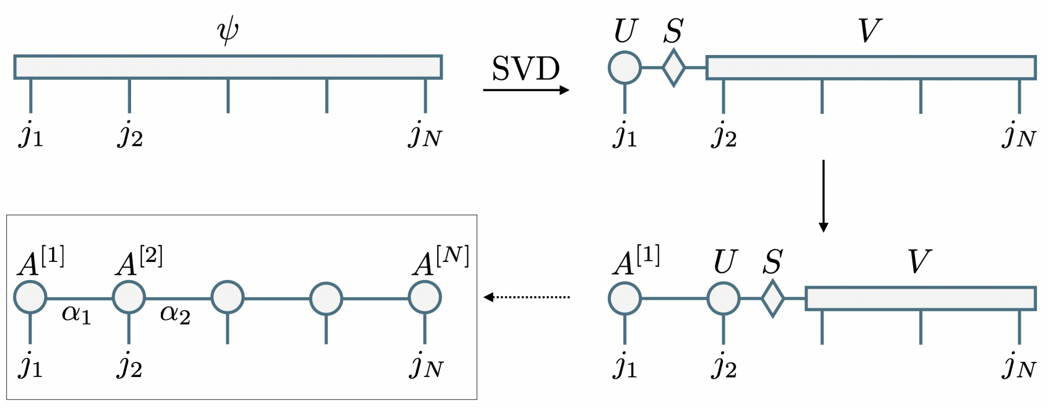


Figure 1.6: Decomposition of a wavefunction ψ (upper left) into a MPS (lower left). Tensors are split off the original wavefunction by performing successive singular value decompositions. The singular value matrix S can be absorbed into the adjacent tensor.

splitting off the tensor at site 2. We repeat this procedure until we have reached the final site N. The diagonal singular value matrices can be absorbed into either of the two neighboring tensors. This leaves us with the MPS representation (see box in Fig. 1.6)

$$\psi_{j_1,\dots,j_N} = \sum_{\alpha_1,\dots,\alpha_{N-1}} A^{[1]j_1}_{\alpha_1} A^{[2]j_2}_{\alpha_1\alpha_2} \dots A^{[N-1]j_{N-1}}_{\alpha_{N-2}\alpha_{N-1}} A^{[N]j_N}_{\alpha_{N-1}}. \tag{1.16}$$

Each tensor belongs to one site i of the chain and has one physical index j_i as well as one or two virtual indices α_i, α_{i+1} that are summed over. The dimension of the virtual indices are determined by the SVD and called bond dimension χ. Assuming a uniform bond dimension χ for all indices α_i we find that the number of parameters in the MPS ansatz scale roughly as $Nd\chi^2$. Hence, for a fixed bond dimension χ the memory requirements scale only linear in the system size N as opposed to the exponential scaling we observe for the full rank-N wavefunction.

MPS can also be considered as a variational ansatz for quantum states. The number of variational parameters are determined by the bond dimension χ which is related to the maximum amount of entanglement that can be shared across a bond between two subsystems. Consider for example the von Neumann entanglement entropy between an arbitrary bipartition A/B of the MPS which is defined in terms of the reduced density matrix $\rho_A = \mathrm{Tr}_A \left|\psi\right\rangle \left\langle\psi\right|$ of either of the two subsystems

$$S\left(N_A\right) = -\,\mathrm{Tr}\,\rho_A \log \rho_A \leq \log(\chi). \tag{1.17}$$

The entanglement entropy can be expressed using the eigenvalues s_α of the reduced density matrix, i.e., the singular values, and is therefore upper bounded by the logarithm of the bond dimension χ. Thus, the bond dimension can be chosen independent of the subsystem size N_A as long as the entanglement entropy does not scale with the subsystem size. The quantum states fulfilling this condition are the so called area-law entangled states in 1d and provide the class of states for which an MPS description is indeed efficient [123, 124]. Interestingly, this class is equivalent to the family of ground

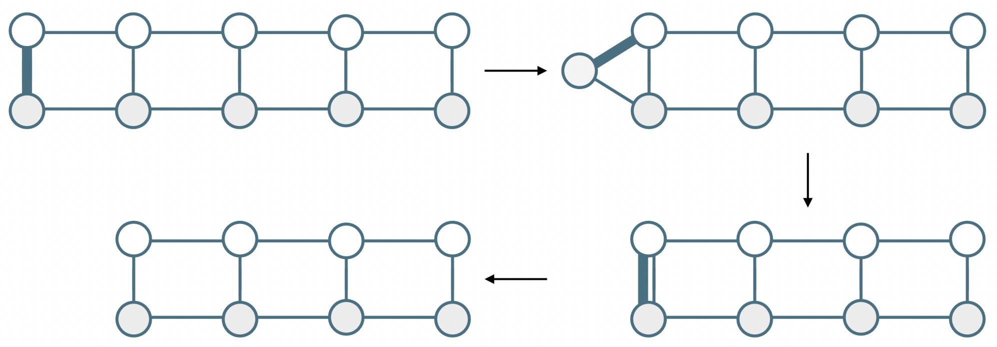


Figure 1.7: Steps for calculating the overlap of two MPS states efficiently. The indices which are contracted at each step are highlighted by thick lines. We first contract over the physical index and then over the two virtual ones giving rise to the diagram on the lower left. We can then repeat this scheme until the network is fully contracted.

states of local gapped Hamiltonians [125]. Note that most states of the many-body Hilbert space do not fall into this category and instead follow a volume law of entanglement, that is, the entanglement entropy is extensive and hence grows linearly with the system size.

The bond dimension of an MPS, and with that the entanglement, can be controlled via performing SVDs across bisections and truncating the singular value matrix by throwing away the smallest entries. Hence, we can also regard the MPS as a compression of the quantum many-body state where the level of compression is determined by the bond dimension. Equivalently, we can also consider the bond dimension as the quantity controlling the expressivity of an MPS ansatz similarly to the hidden dimension of a neural network. As long as the bond dimension is chosen sufficiently large, an MPS can represent any quantum state or tensor.

Let me also briefly illustrate how the overlap $\langle\psi|\phi\rangle$ of two MPS wavefunctions can be efficiently computed with time and memory resources scaling only linearly in the system size N. Calculating overlaps of two MPS is an important subroutine of many MPS-based algorithms. The overlap corresponds to the contraction of two MPS over all virtual and physical indices as shown in Fig. 1.7. We start at one side and first contract over the physical index followed by the two virtual indices. The result is a tensor network diagram shortened by one site for which we can repeat the steps above until we fully reduced it to a scalar.

Finally, MPS are not only efficient in terms of their memory usage, but also many relevant computations on quantum states can be performed efficiently within the MPS framework, such as the overlap computation as outlined above. Furthermore, there exist efficient MPS-based algorithms for calculating ground states of (local) Hamiltonians (e.g. DMRG) [36, 126] and for time evolving quantum states (e.g. TEBD, TDVP) [118, 127, 128]. These algorithms feature a linear complexity scaling with the system size. This favorable scaling make MPS therefore a powerful numerical framework for classically simulating 1d quantum many-body systems well beyond exact diagonalization regimes ($N > 20$).

As a final remark, let me note that there is also a straightforward generalization of MPS to operators and density matrices called matrix product operator (MPO) [129].

Furthermore, a variety of tensor networks have been developed for simulating quantum states with different properties such as the MERA for critical states [118], or PEPS for systems defined on higher dimensional lattices [130].

1.3.3 Tensor network-based machine learning

Similarly to how tensor networks can be used as variational ansatze for quantum states, we can employ them as trainable models for machine learning tasks. Tensor networks can be used to compress the weights of neural networks providing a form of regularization [34], added to existing learning architectures creating hybrid ansatze [35], or used completely in place of neural networks [33]. In the following, I will introduce the latter of these approaches in more detail.

One of the early applications of tensor networks to machine learning was by Stoudenmire and Schwab [33]. They considered the typical MNIST example of classyfing handwritten digits. In analogy to support vector machines and kernel methods, the model is chosen to be linear and represented by a single weight matrix W. However, it is defined in a high-dimensional embedding space of the input images $\mathbf{x}$, described by the map Φ

$$f^{\ell}(\mathbf{x}) = W^{\ell} \cdot \Phi(\mathbf{x}), \tag{1.18}$$

where ℓ is the index for each of the 10 classification labels. While the linearity of the ansatz would greatly limit the expressivity, the model receives its power from the nonlinear embedding of the input images in the higher dimensional feature space. Different mappings Φ are possible. In the original work each grayscale pixel x_i is mapped to a spin-1/2 degree of freedom (i.e., a vector) via

$$\phi^{s_i}(x_i) = \left[\cos\left(\frac{\pi}{2}x_i\right), \sin\left(\frac{\pi}{2}x_i\right)\right], \tag{1.19}$$

such that white pixels result in a spin-up state, black pixels in the spin-down state and intermediate pixel values in a superposition thereof. Applying this feature map to one full image $\mathbf{x}$ gives rise to a product state of differently oriented spin-1/2 particles

$$\Phi^{s_1 s_2 \cdots s_N}(\mathbf{x}) = \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \phi^{s_N}(x_N). \tag{1.20}$$

The weight matrix $W^{\ell}_{s_1 s_2 \cdots s_N}$ maps from the 2^N dimensional feature space to the 10 dimensional label space and can be regarded as a $(N+1)$-rank tensor. Hence, the number of parameters in the ansatz and the complexity of evaluating it scale exponentially in the input image size N. Therefore, we instead represent the weight matrix as a matrix product state (see Fig. 1.8)

$$W^{\ell}_{s_1 s_2 \cdots s_N} = \sum_{\alpha_1,\ldots,\alpha_{N-1}} A^{\alpha_1}_{s_1} A^{\alpha_1\alpha_2}_{s_2} \cdots A^{\ell;\alpha_i\alpha_{i+1}}_{s_i} \cdots A^{\alpha_{N-1}}_{s_N}. \tag{1.21}$$

Note that the index ℓ corresponding to the output label vector has to be attached to one of the tensors in the ansatz. However, the label index can be moved to any other position in the MPS via performing SVDs.

For simplicity the cost function C is chosen to be the mean squared error between

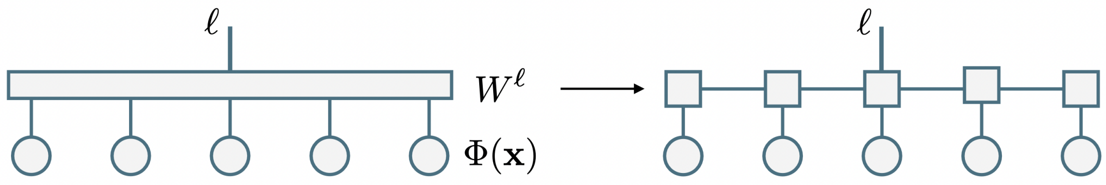


Figure 1.8: Tensor network representation of the machine learning model used in Ref. [33]. The input image $\mathbf{x}$ is first mapped to a product state of spin-1/2 particles via a nonlinear map $\Phi(\mathbf{x})$ and then contracted with the weight matrix W^ℓ resulting in a vector of classification probabilities. Representing the weight matrix as an MPS gives rise to the tensor network diagram on the right side. Here, the trainable parameters (tensors) are denoted by squares to differentiate them from the input (circles).

the function output and the one-hot encoding of the classification labels L_n

$$C = \frac{1}{2}\sum_{n=1}^{N_T}\sum_{\ell}\left(f^\ell\left(\mathbf{x}_n\right) - \delta^\ell_{L_n}\right)^2 . \tag{1.22}$$

The loss function can be minimized using conventional gradient descent or any of its variants and the gradients with respect to each tensor in the MPS can be computed via backpropagation. However, the linearity of the MPS ansatz also allows for other optimization routines involving only local updates which are motivated by the density matrix renormalization group (DMRG) algorithm [36, 126]. Here, at each step of the optimization we only consider one or two tensors at a time and keep all others fixed. We then calculate the gradient with respect to the tensor and perform a gradient descent step. Alternatively, one can rephrase the cost function optimization in terms of a linear equation problem which can be solved via fast numerical linear-algebra routines. Note that the latter approach is only possible since the model is linear in each of its parameters (tensors). After the tensor has been updated, the steps above are repeated for an adjacent tensor. In this way, the full ansatz is optimized by sweeping back and forth through the MPS. If two tensors are updated in every step, we first contract them into one tensor, update the resultant composite one, and then split it again via an SVD. The advantage of the two-site update is that the bond dimension can be adapted as part of the optimization algorithm, i.e., we can set a truncation threshold below which singular values are automatically truncated. This gives rise to an optimization routine which not only updates the model parameters, but also optimizes the overall number thereof.

The use of MPS for supervised learning tasks like MNIST have been further explored in several works [38–41]. Similarly, MPS can also be leveraged for unsupervised learning such as generative modeling [42–44] or anomaly detection [131]. Different training schemes for MPS have been investigated in Refs. [132, 133]. It was shown that gradient-based optimization of MPS can suffer from so called Barren plateaus, i.e, the gradients vanish exponentially in the system size similarly to the training of parameterized quantum circuits [132–134]. MPS, or more specifically its operator analog MPO, have also been employed for compressing weight matrices in conventional neural networks [34]. Here, each weight matrix is first reshaped into a higher rank tensor and then expressed as an MPO using far fewer parameters. The achievable test set accuracies were on par with uncompressed neural network architectures, however, often allowed for faster convergence. MPS have also been leveraged as a feature extractor or dimensional reduction technique for classical data in a quantum machine learning setting [35, 135, 136]. The data is first mapped to a small feature space and then encoded into qubits followed by the application of a parameterized quantum circuit. The MPS and the quantum circuit are optimized end-to-end giving rise to a hybrid architecture which has been applied both to a classification [35, 135] and a reinforcement learning problem [136].

Similarly to MPS, we can also use other tensor networks as parameterized machine learning ansatze. A natural extension of MPS to two dimensional systems are PEPS which have already been applied to the typical MNIST classification task [137]. Moreover, TTN and the MERA have been investigated as ansatze and shown to give superior performance to MPS in many tasks likely due to their ability of capturing long-range correlations more efficiently [138–142]. Other generalized tensor network architectures such as locally purified states or string-bond states have also been previously considered for machine learning [143, 144]. Note that there also exists a connection between tensor networks and some neural network architectures [145–147]. For example, it was shown that MPS and restricted Boltzmann machines (RBM) can be mapped onto each other [147].

1.4 Quantum machine learning

In this final section, I turn to machine learning models that are quantum, i.e., they are (partly) represented and evaluated on quantum devices. The motivation for these quantum machine learning (QML) algorithms is two-fold: On one hand, we hope that using quantum resources allows for a substantial better scaling of the model and training complexities similarly to how some quantum algorithms can gives rise to exponential speedups over their classical counterparts. On the other hand, we know that there are non-classical patterns that quantum systems can efficiently represent and that could potentially be leveraged by QML. Consequently, quantum machine learning models might perform better (achieve higher accuracies) than classical machine learning methods. Note that none of these promising advantages has yet been demonstrated in real-world examples.

In the following, I will first introduce the circuit model of quantum computation, give a general overview of quantum machine learning approaches, and discuss variational quantum algorithms as a specific example. General introductions to the field of quantum machine learning can be found in Refs. [148–151].

1.4.1 Gate-based quantum computing

The basic entity storing the information of a quantum computation is the qubit [99]. A qubit is a two-level system like a spin-1/2 degree of freedom and its state is fully specified by two complex numbers a and b

$$|\psi\rangle = a|0\rangle + b|1\rangle. \tag{1.23}$$

Since the quantum state has to be normalized, $\langle\psi|\psi\rangle = 1$, we require that $|a|^2+|b|^2 = 1$. Alternatively, we can describe the qubit via three, real-valued angles γ, θ, ϕ

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right), \tag{1.24}$$

where γ is a global phase and therefore often omitted. The angles θ and ϕ can be interpreted as the polar and azimuthal angles of a unit vector. Hence, a pure single-qubit state represents a point on the surface of a unit sphere in 3d, referred to as the Bloch sphere. The computational basis states $|0\rangle$ and $|1\rangle$ correspond to the north-and southpoles of the Bloch sphere respectively, while the uniform superposition $|\psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle)$ lies along the equator.

The composite state of several qubits lives in the tensor-product Hilbert spaces of each constituent qubit, i.e., $|\psi\rangle \in \mathbb{C}^{2^n}$. Hence, we again encounter the curse of dimensionality if we were to store the 2^n-dimensional n-qubit state on a classical computer. Instead, on an ideal quantum computer the information is inherently stored and processed using only n qubits. A general quantum computation is represented by a quantum circuit which can roughly be decomposed into three steps: qubit initialization, the application of unitary operators (so called quantum gates), and the final measurement[*]. Depending on the algorithm these steps have to be repeated and are followed by classical post-processing of the measurement statistics.

By convention, qubits are always initialized in their "ground state" $|\psi\rangle = |0\rangle^{\otimes n}$. The qubit state is then altered by applying unitary gates U_i that, in principle, can act on all qubits $|\psi'\rangle = U_L \cdots U_2 U_1 |\psi\rangle$. The unitary operators that have to be applied depend on the respective algorithm being implemented. Similarly to the finite set of classical logic gates realizable on classical computers, a physical quantum computer will not be able to natively perform arbitrary quantum gates. Instead, only a subset of operations will be supported and an arbitrary unitary first has to be decomposed into the corresponding native gate set of the physical device. It has been previously shown that any (global) unitary can be decomposed into only two-and single-qubit gates [152, 153]. Furthermore, there are a number of different gate sets that allow for universal quantum computation [99]. For example, the CNOT (CX), H, S, and T gates defined below constitute such a universal set. The Hadamard gate H, phase gate S, and T gate are single-qubit gates and in the computational basis their matrix representation is given by

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}. \tag{1.25}$$

For instance, the Hadamard gate H maps a computational basis state $|0\rangle$ to a uniform

[*]It is also possible to perform mid-circuit measurements where the result is used in the subsequent quantum computation. Hence, it might also be necessary to repeatedly reinitialize qubits as part of the computation.

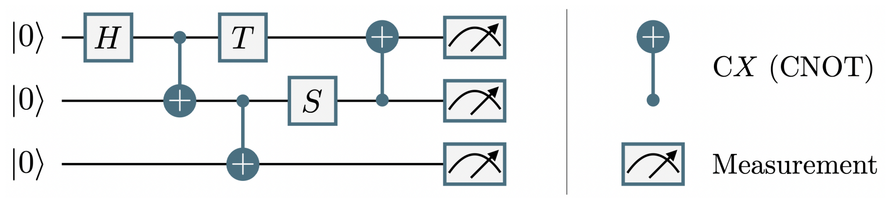


Figure 1.9: A circuit diagram of an exemplary quantum circuit of three qubits involving Hadamard gates H, phase gates S, T gates, and CX (CNOT) gates. A line (or wire) represents a qubit which is usually initialized in the $|0\rangle$ state. Circuit operations are read left to right. At the end of the circuit all qubits are measured in the computational basis.

superposition state $|+\rangle \equiv \frac{1}{2}(|0\rangle + |1\rangle)$. The controlled-NOT or CX gate is a two-qubit gate defined by

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.26}$$

It applies a bit-flip (Pauli-X gate) to the second qubit (target qubit) conditional on the first qubit (control qubit) being in the $|1\rangle$ state. In general, a two-qubit unitary like the CX gate entangles the qubits it acts on.

Apart from the examples above, another widely-used operation realized in many current quantum devices are parameterized rotation gates

$$R_P(\theta) = \exp(-i\theta P/2) = \cos(\theta/2)I - i\sin(\theta/2)P, \tag{1.27}$$

which define a rotation with angle θ about an axis P on the Bloch sphere. The generator P of the rotation is usually given by one of the Pauli operators $P = \{X, Y, Z\}$.

Finally, to obtain the result of a quantum computation we require one or more qubits to be measured. Measurements are usually projective and performed in the computational basis. Hence, the probability of measuring a specific bitstring m given the pure quantum state $|\psi\rangle$ is $p(m) = \langle\psi|P_m|\psi\rangle$, where P_m is the projector onto the computational basis state $|m\rangle$. After the state has been measured, it is destroyed, i.e., the state has been projected onto the corresponding outcome $|m\rangle$. Often one is interested in measuring probabilities or expectation values of observables rather than single bitstrings. In this case, we have to repeatedly run the quantum circuit and average over the measurement outcomes. If measurements of operators other than Pauli-Z are required, we additionally have to perform basis transformations before the measurement. We can represent a given quantum computation defined by its gates and measurements via a circuit diagram as shown in Fig. 1.9.

1.4.2 Quantum machine learning and NISQ

Quantum machine learning has become an ubiquitous term and nowadays encompasses a variety of different ideas and approaches. First of all, I will use the term quantum machine learning whenever a part of the machine learning algorithm is substituted by a quantum computation (irrespective of whether the data is classical or quantum). Hence, the case of using fully classical machine learning techniques for quantum data – also sometimes referred to as QML in the literature – is not discussed in the following, but will be the content of Section 1.5.

Originally, QML referred to the usage of specific quantum algorithms for speeding-up certain linear algebra operations in classical machine learning techniques [149]. Methods like gradient descent, Newton's method, principal component analysis, or support vector machines can be sped up using quantum algorithms such as amplitude amplification, phase estimation, the HHL algorithm (matrix inversion), and quantum random access memory [154–156]. A different approach to quantum machine learning is to "quantize" classical machine learning models and hence, to fully store, train, and evaluate them using quantum systems. For example, there already exists proposals for nonlinear quantum perceptrons, which could represent the basic building blocks of quantum neural networks [157–160]. Moreover, some architectures such as Boltzmann machines have a straightforward generalization to the quantum case [161].

What most of the approaches above have in common is that they require large-scale, fault tolerant quantum computers to solve the specific problem and provide any gain (speed-up) over the classical algorithms. However, we are currently in the noisy-intermediate scale quantum (NISQ) era, that is, current state-of-the-art quantum computers are subject to various sources of noise and consist on the order of only 100 physical qubits [49]. The errors are still too large and the system sizes too small in order to harness quantum error correction strategies that would suppress errors to arbitrary degrees. Instead, physicists devise heuristic error mitigation strategies to reduce the effects of noise and develop algorithms tailored to NISQ devices which let us explore their current capabilities [45, 46].

Similarly, different quantum machine learning approaches have been explored on NISQ computers [45–47]. One prominent example for supervised classification tasks is quantum kernel estimation [50, 162, 163]. In kernel methods, the input data $\mathbf{x}$ is first mapped to some higher dimensional feature space $\Phi(\mathbf{x})$ and then classified via a decision boundary, i.e., a hyperplane [164]. The classification result $y(\mathbf{x}) = \pm 1$ for an input $\mathbf{x}$ can be expressed as a linear combination of kernels $\kappa(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, that is, inner products of feature maps in the high dimensional space

$$y(\mathbf{x}') = \operatorname{sign} \sum_{i=1}^{n} w_i \hat{y}_i \kappa(\mathbf{x}_i, \mathbf{x}'), \tag{1.28}$$

where w_i are parameters to be optimized and $\hat{y}_i$ is the known label for data point $\mathbf{x}_i$. The linearity of the ansatz gives rise to a convex optimization problem in the coefficients w_i.

In quantum kernel estimation, the input data $\mathbf{x}$ is instead mapped to a quantum state $|\Phi(\mathbf{x})\rangle$ in a Hilbert space. A quantum circuit is then used to compute the inner products of different feature maps $\kappa(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}) | \Phi(\mathbf{x}') \rangle$. Finally, the extracted kernels are fed into an optimization routine on a classical computer.

A related class of NISQ algorithms that can be used for near-term QML are variational quantum algorithms (VQA) [45, 46, 48]. These methods are explained in detail in the next section. In a nutshell, they also require the classical data to be first encoded into a quantum state $|\Phi(\mathbf{x})\rangle$. However, instead of computing kernels and calculating Eq. (1.28), we use a parameterized circuit $U(\boldsymbol{\theta})$ as the model and directly calculate the output in the quantum Hilbert space. Training the model then amounts to optimizing the quantum circuit itself, i.e., its parameters $\boldsymbol{\theta}$ similarly to a classical neural network ansatz [165, 166].

1.4.3 Variational quantum algorithms

Variational quantum algorithms (VQA) describe a class of quantum algorithms which are all based on the variational optimization of parameterized quantum circuits [45–48]. VQAs can be employed for a variety of different problems including ground state search, Hamiltonian simulation, and quantum machine learning. In the following, I will introduce VQAs from a machine learning perspective in which case they are also sometimes referred to as variational quantum classifiers or quantum neural networks.

At the core of VQAs lies a parameterized quantum circuit $U(\boldsymbol{\theta})$ which can be interpreted as the model that has to be trained. The model output is computed by first applying the unitary $U(\boldsymbol{\theta})$ to the input state $|\Phi(\mathbf{x})\rangle$

$$|\Psi(\mathbf{x},\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\Phi(\mathbf{x})\rangle\,, \tag{1.29}$$

and then measuring the resultant state $|\Psi(\mathbf{x},\boldsymbol{\theta})\rangle$. The choice of cost function depends on the problem, but is usually written in terms of a fidelity or expectation value of the final state $|\Psi(\mathbf{x},\boldsymbol{\theta})\rangle$. For example, in the case of a classification task where labels $\hat{y}_i$ have been encoded into some quantum states $|\hat{y}_i\rangle$, the cost function C can be expressed in terms of the infidelity of the target state $|\hat{y}_i\rangle$ and the circuit output $|\Psi(\mathbf{x}_i,\boldsymbol{\theta})\rangle$

$$C(\theta) = \sum_{i=1}^{N_T} \left[1 - |\langle y_i|\Psi\left(\mathbf{x}_i,\boldsymbol{\theta}\right)\rangle|^2\right]. \tag{1.30}$$

While the model is evaluated on a quantum device, the optimization routine for the model parameters $\boldsymbol{\theta}$ is performed on a classical computer. Therefore, VQAs are also often called hybrid quantum-classical algorithms (see Fig. 1.10).

The choice of the circuit ansatz $U(\boldsymbol{\theta})$ depends on various factors such as the problem we are trying to solve and hardware limitations. On one hand, we would like to make the circuit ansatz as expressive as possible by using deep circuits, all-to-all connected entangling gates, and a large number of parameters. However, with current NISQ devices, the state fidelities quickly deteriorate with growing circuit depths due to noise. Moreover, most quantum hardware currently have only very limited qubit connectivities (e.g. nearest neighbor connections on a heavy hex lattice as is the case for the IBM Quantum devices). Hence, entangling arbitrary qubits that are not adjacent on the device requires several SWAP operations which artificially increases the depth of a circuit and with it the introduced errors.

A common choice for the circuit ansatz $U(\boldsymbol{\theta})$ that addresses this trade-off between expressivity and hardware constraints are so called hardware-efficient ansatze [167]. They are typically comprised of a very limited gate set including a single two-qubit gate (e.g. the CX or CZ gate) and a few parameterized single-qubit gates (i.e. R_x, R_y, R_z

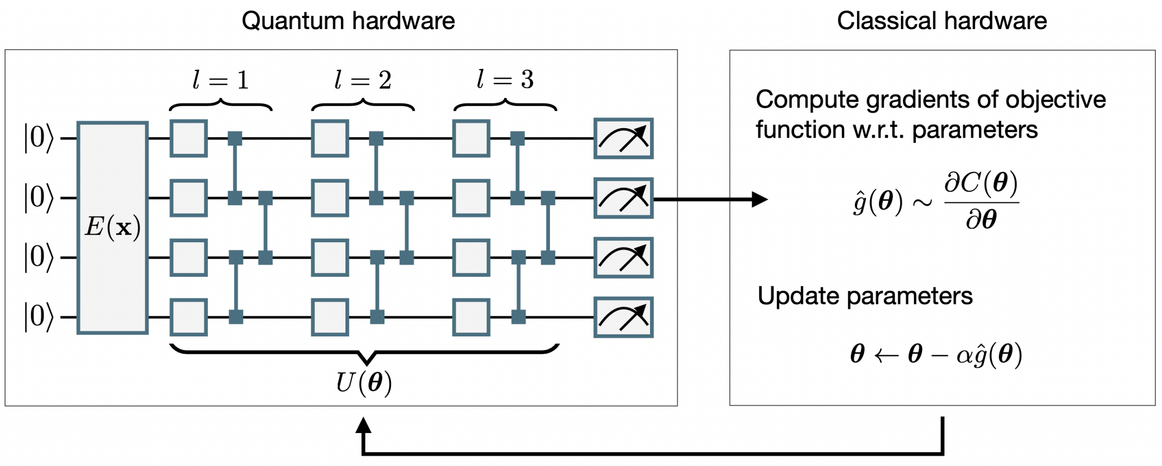

Quantum hardware
Classical hardware
$l=1$
$l=2$
$l=3$
$|0\rangle$
$E(\mathbf{x})$
$U(\boldsymbol{\theta})$
Compute gradients of objective function w.r.t. parameters
$\hat{g}(\boldsymbol{\theta}) \sim \frac{\partial C(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
Update parameters
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \hat{g}(\boldsymbol{\theta})$

Figure 1.10: The quantum-classical feedback loop of variational quantum algorithms. Left box: The machine learning model is represented by a parameterized quantum circuit $U(\boldsymbol{\theta})$ which can be decomposed into several layers l of identical sub-circuits. In the hardware efficient ansatz, each layer l usually consists of a block of parameterized single-qubit rotation gates (squares) and fixed entangling gates (lines connecting different wires). Classical data first has to be mapped to a quantum state on the circuit, for example by applying an input-dependent encoding unitary $E(\mathbf{x})$. The final output of the quantum circuit is measured and aggregated into an average quantity like a fidelity or expectation value. Right box: The optimization of the parameters $\boldsymbol{\theta}$ is performed on classical hardware via minimizing a cost function C. The updated parameters are then used in the next evaluation of the circuit.

gates). The circuit is then built up from several identical layers l of smaller circuit blocks that each consist of single qubit gates $R_P(\theta_{l,i})$ applied once to all qubits $i = 1, \ldots, n$, and an entangling unitary W_l. The entangling unitary W is, in turn, decomposed into several two-qubit gates that are usually applied to only adjacent qubits on the physical device (see Fig. 1.10). Hence, in total we can express the circuit ansatz $U(\boldsymbol{\theta})$ as

$$U(\boldsymbol{\theta}) = \prod_{l=1}^{L} U_l(\boldsymbol{\theta}_l) W_l = \prod_{l=1}^{L} \left(\bigotimes_{i=1}^{n} R_P(\theta_{l,i}) \right) W_l, \tag{1.31}$$

where L denotes the total number of layers. Note that the optimizable parameters $\boldsymbol{\theta}$ enter only through the single-qubit rotations while the two-qubit entangling gates are generally fixed. Furthermore, we usually allow the parameters $\theta_{l,i}$ to be different for each qubit i and layer l. I discuss the explicit optimization methods for the circuit parameters further below.

When employing VQAs for machine learning on classical data we also have to specify a data-to-qubit mapping. Different qubit encodings have been previously proposed [50, 150, 162, 168]. An efficient encoding for current NISQ devices is the data uploading approach in which the input data $\mathbf{x}$ is mapped to the qubit state $|\Phi(\mathbf{x})\rangle$ via an input-dependent unitary $E(\mathbf{x})$

$$|\Phi(\mathbf{x})\rangle = E(\mathbf{x}) |0\rangle . \tag{1.32}$$

The unitary $E(\mathbf{x})$ can be constructed similarly to the circuit ansatz $U(\boldsymbol{\theta})$ of Eq. (1.31), i.e., it consists of several layers l of parameterized, single-qubit rotations $R(x_{l,i})$ and entangling gates W_l

$$E(\mathbf{x}) = \prod_{l=1}^{L} U_l(\mathbf{x}) W_l. \tag{1.33}$$

Here, L_E is the total number of layers of the data encoding circuit. It is also possible to introduce another set of parameters $\boldsymbol{\phi}$ into the data uploading unitary $E(\mathbf{x}, \boldsymbol{\phi})$ and hence, also optimize over the qubit mappings. Alternatively, one can also merge the two parts of data encoding and trainable model into a global circuit $U(\mathbf{x}, \boldsymbol{\theta})$ where data uploading and parameterized layers are alternated [169]. This scheme, referred to as data reuploading, has been proven universal for classification and shown to give rise to a Fourier series in the data [169, 170].

The final component of VQAs is the classical optimization of circuit parameters $\boldsymbol{\theta}$. VQAs give rise to a standard multivariate optimization problem for which we can use any classical optimization method such as gradient descent. Gradient-based approaches require us to estimate the partial derivatives of the circuit output $f(\mathbf{x}; \boldsymbol{\theta})$ (e.g. an expectation value or a fidelity) with respect to each parameter θ_i. If the parameters in the circuit $U(\boldsymbol{\theta})$ enter solely via the single-qubit rotation gates $R_P(\theta_i)$, we can compute the gradients exactly using the parameter-shift rule [165, 171]

$$\frac{\partial f(\mathbf{x}; \theta_i)}{\partial \theta_i} = \frac{1}{2}\left[f\left(\mathbf{x}; \theta_i + \frac{\pi}{2}\right) - f\left(\mathbf{x}; \theta_i - \frac{\pi}{2}\right)\right]. \tag{1.34}$$

Hence, the gradients can be expressed via the same variational circuit $U(\boldsymbol{\theta})$ albeit shifted parameter values. Each gradient calculation therefore requires two "forward passes", i.e., evaluations of the objective function $f(\mathbf{x}; \boldsymbol{\theta})$. Note, however, that the objective function usually represents an expectation value or a fidelity and therefore we need to collect, in principle, a large number of circuit measurements to estimate $f(\mathbf{x}; \boldsymbol{\theta})$ with high enough precision. The training of quantum circuits using exact gradients is therefore considerably less efficient than the training of classical neural networks. The latter involves only a single forward and backward pass (both of which have the same complexity scaling) to compute the gradients with respect to all neural network parameters. Moreover, with the use of parallelization techniques or modern GPUs, the gradient computation can be efficiently parallelized over the whole input data. This stands in contrast to the training of VQAs where the number of forward passes (circuit evaluations) scales linearly with the number of parameters and the training data size.

Due to these reasons, other optimization strategies for parameterized quantum circuits have been explored [46]. One notable example is the simultaneous perturbation stochastic approximation (SPSA) algorithm [172]. SPSA is a stochastic method that computes approximate gradients $\hat{g}(\mathbf{x}; \boldsymbol{\theta})$ which are then used as a replacement for the exact gradients in the conventional gradient descent update rule. SPSA estimates the gradients via a modified finite difference method: instead of computing the finite differences separately for each parameter, we sample a random perturbation vector $\boldsymbol{\Delta}$ and apply it to all parameters simultaneously

$$\hat{g}\left(\mathbf{x};\boldsymbol{\theta}\right)_i = \frac{f\left(\mathbf{x};\boldsymbol{\theta}+c\boldsymbol{\Delta}\right) - f\left(\mathbf{x};\boldsymbol{\theta}-c\boldsymbol{\Delta}\right)}{2c\Delta_i}, \tag{1.35}$$

with c being a small positive number. Estimating the gradients via this method has two major advantages. First, each optimizer step now only requires two function evaluations in order to obtain the gradients for all parameters. Hence, the number of required circuit executions per iteration does not scale with number of parameters. Second, since the gradient computation is based on stochastic perturbations, the performance of SPSA can be robust to the noise and stochasticity introduced by physical devices [50, 167].

Despite the issues of efficient trainability, VQAs currently face an additional challenge regarding their optimization, the so called Barren plateaus phenomenon [173]. Barren plateaus arise when the expectation value of the gradient and its variance vanish exponentially with the circuit depth or the number of qubits. In this case, the loss function landscape is mostly flat and interspersed with many narrow gorges of local minima [173]. The appearance of Barren plateaus is related to the curse of dimensionality, i.e., the exponentially growing Hilbert space dimension as the number of particles is increased. Several approaches have been proposed for alleviating the Barren plateau problem [45, 46]. For example, it has been shown that using local cost functions instead of global ones [174], using layerwise optimization strategies for the circuit [175], and choosing initial parameters that are already close to the optimum [176] can each help in reducing the occurrence of Barren plateaus.

1.4.4 Quantum autoencoder

The quantum autoencoder (QAE) is a specific example of a variational quantum algorithm and generalizes the idea of classical autoencoders to the quantum case [177]. Thus, the goal is to first compress a set of quantum states $\{|\psi_i\rangle\}$ into a low-dimensional latent space representation, and then to reconstruct the original states again given just the latent state information.

Similarly to the classical AE, the quantum circuit representing the QAE can be separated into an encoder and decoder part. The encoder circuit consists of a parameterized unitary $U_E(\boldsymbol{\theta})$ that is applied to an input quantum state $|\psi\rangle$ defined on n qubits. Since all quantum operations are unitary, the dimensionality of the qubit state after the encoder circuit will be unchanged. Hence, we additionally need to trace out (or measure) part of the system to effectively reduce its dimension. After the information has been encoded onto a smaller number of k qubits, we apply the decoder circuit $U_D(\boldsymbol{\theta})$. To match the dimensions of the original state again, the decoder unitary is applied to a product state of the latent qubits and additional ancilla qubits (i.e. qubits initialized in the $|0\rangle^{\otimes(n-k)}$ state). The final state ρ_{out} at the output of the QAE can be expressed as

$$\rho_{\text{out}}(\boldsymbol{\theta}) = U_D(\boldsymbol{\theta}) \left[\text{Tr}_T\left(U_E(\boldsymbol{\theta})\left|\psi\right\rangle\left\langle\psi\right| U_E^\dagger(\boldsymbol{\theta})\right) \otimes \left|0\right\rangle\left\langle 0\right|^{\otimes(n-k)}\right] U_D^\dagger(\boldsymbol{\theta}), \tag{1.36}$$

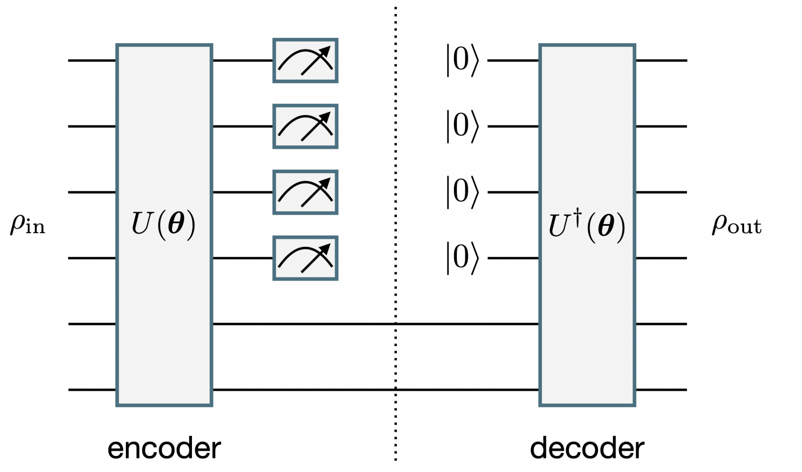


Figure 1.11: Circuit diagram of the quantum autoencoder. The information of the input state ρ_{in} is mapped to a low-dimensional latent representation by first applying the encoder unitary $U(\boldsymbol{\theta})$ and then measuring a subset of qubits, called trash qubits. The decoder reconstructs the input state again by applying the inverse $U^\dagger(\boldsymbol{\theta})$ of the encoder unitary to the product state of latent qubits and ancilla qubits (i.e. qubits initialized in the all-zero state). An ideal autoencoder is able to perform lossless compression such that $\rho_{\text{out}} = \rho_{\text{in}}$ for all data inputs. When training the autoencoder it is sufficient to evaluate only the encoder part of the circuit up to the dashed vertical line.

where $\text{Tr}_T(\cdots)$ denotes a partial trace over the $n-k$ "trash" qubits that are discarded and $\rho_{\text{in}} = |\psi\rangle\langle\psi|$ is the density matrix of the input state.

Due to the unitary property of quantum computing, lossless compression (that is $\rho_{\text{out}} = \rho_{\text{in}}$) can only be achieved if the encoder maps all input states $|\psi_i\rangle$ to states $|\psi_i'\rangle \otimes |a\rangle$, where $|a\rangle$ is an input-independent reference state. For simplicity, we can take the reference state to be the all-zero state $|a\rangle = |0\rangle^{\otimes(n-k)}$. In this case, it is easy to see that the decoder circuit $U_D(\boldsymbol{\theta})$ has to be the inverse of the encoder circuit, i.e., $U_D(\boldsymbol{\theta}) = U_E^\dagger(\boldsymbol{\theta})$ in order to always fulfill the requirement that $\rho_{\text{out}} = \rho_{\text{in}}$. Hence, from now on I will omit the subscripts. The circuit diagram of the quantum autoencoder is depicted in Fig. 1.11.

The ansatz for the parameterized QAE unitary $U(\boldsymbol{\theta})$ can be chosen as for any other variational quantum algorithm. Training the QAE then amounts to optimizing the shared encoder and decoder parameters $\boldsymbol{\theta}$. A natural figure of merit we can minimize is the average infidelity between the in-and output states defined by

$$C_1 = 1 - \frac{1}{N_T}\sum_{i=1}^{N_T}\langle\psi_i|\rho_{\text{out},i}(\boldsymbol{\theta})|\psi_i\rangle. \tag{1.37}$$

An alternative objective would be to optimize the encoder unitary $U(\boldsymbol{\theta})$ such that the trash qubit state is always equal to the all-zero state $|0\rangle^{\otimes(n-k)}$

$$C_2 = 1 - \frac{1}{N_T}\sum_{i=1}^{N_T}\langle 0|\text{Tr}_{\bar{T}}\left(U_E(\boldsymbol{\theta})\,|\psi_i\rangle\langle\psi_i|\,U_E^\dagger(\boldsymbol{\theta})\right)|0\rangle, \tag{1.38}$$

where $\mathrm{Tr}_{\bar{T}}(\cdots)$ now denotes a partial trace over the k qubits in the latent space. The advantage of the second cost function C_2 is that the overall circuit depth for evaluating it is effectively cut in half, since we only have to execute the encoder circuit. In contrast to classical autoencoders, the QAE therefore allows us to only optimize the encoder which then automatically amounts to also improving the decoder. Moreover, the loss function C_2 has the additional advantage that it does not require knowledge of the initial state as it is only defined via the trash qubit fidelities. Hence, it allows us to train autoencoders also on quantum states whose exact form is unknown.

In the original proposal, the QAE has been applied to the compression of ground states of the Hubbard model and molecular Hamiltonians [177]. Since then it has also been used on classical data such as for the compression of images of handwritten digits [178]. Furthermore, the QAE has already been experimentally realized on a photonic quantum device [179].

1.5 Machine learning for quantum data

In this section I will briefly review some past applications of machine learning in the quantum domain that are related to the subsequent chapters. For a more comprehensive and detailed review of the field I refer to Refs. [10–12, 81–83].

The problem of phase classification of (quantum) many-body systems represents one of the most famous applications of deep learning to the physical sciences [13, 14]. In one of the early studies, supervised learning techniques were used to classify ground states of the Ising model and the Toric code [13]. The model was trained on states deep within the two phases and then tested along the whole parameter range. The location of the phase transition was indicated by an abrupt change of the classification probability. van Nieuwenburg et al. [14] used a similar supervised learning approach, however, did not require prior knowledge about the location of the critical point. Instead, they labeled the training data ground states by guessing the point of the phase transition in parameter space. The model is then trained on the (falsely) labeled data set. These steps are repeated by choosing different parameter values for the critical point and relabeling the data set accordingly. The final prediction accuracy of the trained models determines the actual location of the phase transition since the model can classify the phases most accurately if the training data was labeled correctly. Note that in contrast to the first example of Ref. [13], this approach requires to optimize a separate neural network for each different parameter value and hence is considerably more expensive. Supervised learning techniques have been applied to the phase classification of strongly-correlated fermionic systems [180], topological systems [181–183], and systems out-of-equilibrium [184, 185].

Fully unsupervised techniques for phase detection based for example on PCA, adversarial NNs or autoencoders have also been explored [18–20, 64, 186, 187]. In Ref. [18] Kottmann et al. use the idea of anomaly detection to map out the phase diagram of the extended Bose Hubbard model. An autoencoder is trained on ground states of a small corner of the phase diagram. For these training examples, the model learns an accurate latent representation of the input states and hence is able reproduce them at the output. However, when tested on the whole parameter regime, the autoencoder fails in replicating the input and consequently gives rise to a large reconstruction error. Plotting the error across the phase diagram reveals the location and approximate boundaries of all distinct phases. Unsupervised methods for phase discovery have also been applied to study topological [63, 64], frustrated [188], and higher dimensional systems [62].

Another prominent example of a deep learning application in the quantum sciences are neural quantum states (NQS) [15]. Similarly to how tensor networks can be harnessed as variational ansatze to efficiently represent quantum many-body wavefunctions, deep neural networks represent yet another expressive parameterized ansatz that can be leveraged for this task. In one of the first works Carleo et al. [15] used a restricted Boltzmann machine (RBM) to learn the ground states of the transverse field Ising and the antiferromagnetic Heisenberg model as well as non-equilibrium states after quench dynamics. The RBM was trained via the already established variational Monte Carlo method. Neural quantum states have been proven to be more expressive than tensor network-based approaches given the same number of parameters [147, 189]. In particular, it was shown that NQS can efficiently represent volume-law entangled states [190, 191], topologically non-trivial states [192, 193], and frustrated systems [194, 195]. The NQS framework was later generalized to excited states [196], fermionic systems [197], and open quantum systems [198–201]. Furthermore, algorithms for the time evolution of NQS have been developed as well [202, 203].

A related idea to NQS is that of neural-network quantum state reconstruction [21]. Quantum state tomography usually requires computational resources scaling exponentially in the system size both, in terms of storing the state and in terms of the number of measurements. Torlai et al. [21] showed that representing the quantum states as a neural network (specifically an RBM) allowed for a significant reduction of the required number of measurement samples. They demonstrated their framework by reconstructing the highly entangled W state as well as ground states of a Heisenberg model on synthetically generated data. These methods were later also successfully applied to experimental data from a programmable Rydberg quantum simulator [204].

Reinforcement learning (RL) has already been adopted in a variety of different quantum control problems. In one of the first examples [17], a policy-gradient algorithm was used to learn error correction strategies for qubits undergoing decoherence. To that end, the RL agent had to learn protocols composed of gates and measurements that encode the qubit state into a multi-qubit system, measure individual qubits to extract information, and correct errors by applying gates. The RL state was given by a lower dimensional representation of the completely positive map which describes the entire noisy quantum evolution. The reward function represented the recoverable quantum information at each time step. Once the policy network was successfully trained on the full state information, it served as a teacher network to another recurrent neural network which was optimized in a supervised fashion solely on measurement outcomes (rather than the state information which is not available in experiments). Note that RL has also been leveraged in other works for optimizing quantum error correction codes [22–24].

Another promising area of application for RL is that of quantum state preparation which presents a necessary subroutine for many modern quantum technologies. For example, Bukov et al. [16] considered the Q-learning algorithm with linear function approximation to prepare a specific target ground state of the Ising model with transverse and longitudinal fields. The RL state was given by the current time step and the value of the magnetic field. The possible actions were either to increase or decrease the field by a fixed amount. Lastly, the reward at each step was chosen to be zero except for the final state where it equaled the fidelity with respect to the target state. Bukov et al. showed that the optimal protocols devised by the RL agent performed comparably to conventional optimal control techniques. Further, they discovered a spin-glass-like phase transition in the control landscape as a function of the total protocol duration.

Since these early works, RL has been applied to numerous other quantum control problems [29–32, 205–213]. For instance, RL has been used to prepare spin ice states [29], ground states across quantum phase transitions [212], for quantum control scenarios in the presence of noise [30], and for state preparation problems specific to certain quantum simulator platforms [31, 32, 206]. Furthermore, RL has been combined with other control techniques for the purpose of quantum state preparation [211]. Apart from these standard optimal control applications, RL methods have been harnessed for quantum metrology [214, 215] and for quantum circuit compiling, i.e, the task of decomposing global unitaries into the local gate set of a respective quantum device [25–28].

Chapter 2

Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates

The text in this chapter is based on the following publication, but extended to provide additional content.

Friederike Metz, Juan Polo, Natalya Weber, and Thomas Busch
Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates
Machine Learning: Science and Technology **2**, 035019 (2021)

I implemented, trained, and evaluated the machine learning and wrote a first draft of the manuscript. All authors contributed to the discussions and to the editing of the manuscript draft.

2.1 Introduction

Non-equilibrium behaviour of classical and quantum systems is ubiquitous in nature and includes interesting and complex phenomena such as turbulence and chaos, which are still only partially understood [216, 217]. Bose-Einstein condensates (BECs) provide a particularly versatile platform for studying and simulating general features of non-equilibrium dynamics, due to the high level of control over the experimental systems [218, 219]. In particular, rapidly rotating BECs can support quantum vortices, which are considered a key component of superfluid turbulence [53]. Unlike their classical counterparts quantum vortices are restricted to quantized circulation due to the condition that the wave function has to be single valued at all points. This leads to a well-defined velocity profile that is given by the gradient of the phase [220]. Numerous experiments have generated vortices in BECs [221] and observed the formation of vortex-antivortex pairs [222], vortex rings [223], and vortex lattices [224]. Furthermore, in-situ density imaging of vortex cores has opened the door to the analysis of their real-time dynamics [225–227] and thus, the experimental study of chaos, turbulence, and other out-of-equilibrium dynamics [54, 228–231]. For example, recent results include the detection of persistent vortex clusters emerging from the turbulent flow of high-energy vortex configurations [57, 232], the experimental realization of the quantum analogue of the Kármán vortex street [233], and the observation of vortex-antivortex pairing in a turbulent BEC [234].

However, the study of quantized vortices and specifically their dynamics requires to first infer their precise location within a BEC [60]. For ground states the task of detecting vortices is straightforward, since they are arranged in a clear pattern with pronounced density minima at their core centers [224, 235] and therefore can be easily spotted by eye or via automated processes. On the other hand, in non-equilibrium configurations vortices are located at random positions following no distinct order. Furthermore, local density minima not corresponding to vortices can emerge as a consequence of phononic excitations, making the detection of vortices considerably more difficult [236]. In numerical simulations that model the dynamics of BECs one usually has access to the full condensate wave function and hence also to its phase. The phase profile provides a clear indication of the existence of a vortex through a phase winding of 2π around the position of a vortex core. Therefore, vortex detection algorithms for non-equilibrium configurations mainly rely on the BEC phase profile to distinguish vortices from other defects [236, 237]. However, in experiments the phase profile and thus the information encoded therein is not easily accessible. Moreover, non zero temperatures and the presence of noise pose an additional challenge for accurately detecting vortices and hence require the development of more elaborate methods.

In this chapter we show that a machine learning based vortex detector can reliably and accurately locate vortices within out-of-equilibrium BEC density images. It can distinguish vortices from other local density minima even in situations that are difficult for the human eye. In contrast to conventional vortex detection algorithms, such as blob detection, the neural network does not require hard-coded features or fine tuning of parameters [60, 237]. In addition, the model is robust, i.e., it performs well on simulated data with experimentally relevant sources of noise and generalizes to configurations it has not been trained on, which would not be possible with more traditional object detection methods like template matching [238]. Hence, we anticipate that our vortex detector can be broadly employed in experimental studies of non-equilibrium vortex configurations where only the BEC density is accessible. On the other hand, in numerical simulations of the BEC the phase profile is available and can be provided to the neural network as additional information. In this case the model is also able to accurately classify the circulation direction of each vortex.

In recent years machine learning techniques have become a widely adopted tool in the field of quantum physics [12, 83]. Specifically in the area of BECs, machine learning methods have been used to optimize the cooling process for the atomic gas [239], learn the Kosterlitz-Thouless transition [182], and devise control schemes for the creation of quantum vortices [240]. On the other hand, deep learning based object detection has celebrated remarkable successes in the field of classical computer vision, achieving state-of-the-art results in areas like face, vehicle, and medical image detection [241, 242]. Hence, neural network based object detection promises to be a powerful tool for the physical sciences as well and has already been successfully employed in a few cases [243, 244], as for instance to detect and identify characteristics of atomic clouds [245] or to locate dark solitons in a BEC [246]. Finally, let us note that deep learning approaches have also been applied to the detection of vortices in classical fluids such as locating rotor blade tip vortices [247] or eddies in ocean currents [248]. Motivated by these recent successes, in this work we employ a convolutional neural network (CNN) ansatz for the task of vortex detection which can achieve high accuracies on our test data and is therefore very well suited for the problem of locating vortices in BECs.

This chapter is organized as follows: I first present the theoretical model used to simulate BECs and describe how vortices emerge in Section 2.2. Section 2.3 introduces the machine learning based vortex detector. The results of training the model and its ability to generalize to different trapping geometries and different levels of noise are discussed in Section 2.5. Section 2.4 provides further details on the training data, the network architecture, the evaluation metrics, and the features learned by the CNN.

2.2 Physical system

We consider a dilute and weakly-interacting Bose-Einstein condensate rotating around the z-axis with rotational frequency Ω. At zero temperature and assuming a tight harmonic confinement in the z direction such that the transverse dynamics is frozen out, i.e., $\omega_z \gg \omega_\perp$, we can describe the dynamics of the Bose gas in the co-rotating frame by means of the two-dimensional mean field Gross–Pitaevskii equation (GPE) of the form [249, 250]

$$i\hbar\frac{\partial}{\partial t}\Psi = \left(-\frac{\hbar^2}{2m}\nabla^2 + \frac{1}{2}m\omega_\perp^2 r^2 + g|\Psi|^2 - \Omega L_z\right)\Psi, \tag{2.1}$$

with Ψ being the condensate wave function, $\omega_\perp$ the frequency of the harmonic trap, $L_z = xp_y - yp_x$ the angular momentum operator, and $r = \sqrt{x^2+y^2}$ the radial distance. The effective two-dimensional interaction strength is given by $g = g_{3D}/(\sqrt{2\pi}a_z)$ with $a_z = \sqrt{\hbar/m\omega_z}$ and $g_{3D} = 4\pi\hbar^2 a_s/m$ being the harmonic oscillator length scale of the transverse tight confinement and the three-dimensional interatomic interaction strength respectively. Here a_s is the s-wave scattering length. Note that Eq. (2.1) is analogous to the more general nonlinear Schrödinger equation which can describe a variety of different systems [251]. From here onward we use harmonic oscillator units by setting $\hbar = \omega_\perp = m = 1$ and choose interaction strengths $g \in [50, 600]$ as well as rotation frequencies $\Omega \in [0.65, 0.95]$ which correspond to experimentally accessible parameter regimes.

Above a critical rotation frequency Ω_c, the ground state of Eq. (2.1) possesses vortices [252–254]. For large rotation frequencies these vortices arrange themselves in a triangular lattice geometry [224], while for smaller frequencies different configurations can arise [235]. As an example, Fig. 2.1(a)-(b) shows the numerically obtained density distribution $|\Psi(\mathbf{r})|^2$ and phase profile of the ground state wave function when $g = 452$ and $\Omega = 0.816$. The vortices are clearly defined through a density dip at their cores and through the characteristic 2π phase winding in the phase. Note that the detailed structure of the vortex core depends on the trapping potential [219, 220]: in a homogeneous BEC, the width of a vortex core is fixed by the balance between the kinetic and interaction energy, with a typical core size given by the healing length $\xi = \sqrt{8\pi n a_s}$, where n corresponds to the density. In trap systems, the size of the vortex

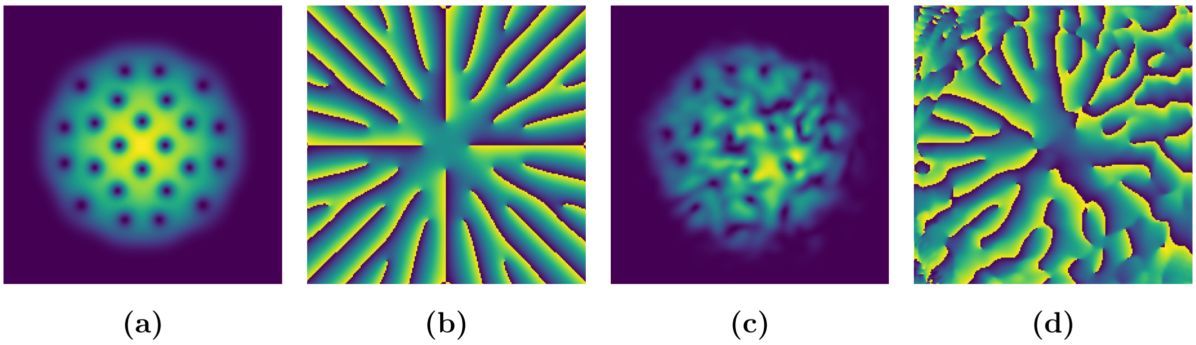


Figure 2.1: Examples of BEC density and phase profiles for the stationary ground state (a),(b) and for a non-equilibrium configuration (c),(d). The ground state is computed via imaginary time evolution with the GPE using an interaction strength $g = 452$ and a rotation frequency $\Omega = 0.816$. Phase imprinting of additional vortices and a subsequent real time evolution gives rise to the out-of-equilibrium configuration.

core depends also on the local chemical potential, which gives rise to slightly larger sizes in low density regions. In addition, vortices surrounded by very low densities at the outer part of the BEC will not contribute to the rotational energy of the system and are therefore irrelevant from a physical point of view [255].

The vortices carried by the ground state all rotate in the same direction, i.e., they have a winding number with the same sign, which is determined by the rotation frequency Ω. Situations where vortices of different rotation directions co-exist can be created for instance by forcing the superfluid to flow around an obstacle potential [256, 257] or through the process of phase imprinting [55, 258–261]. In the latter case, a single vortex centered at (x_0, y_0) is generated by applying a phase mask $\phi_{\mathrm{IMP}}(\mathbf{r}) = \arctan(y - y_0, x - x_0)$ with a 2π phase winding in the desired direction. The time-evolution of configurations with multiple vortices of unequal rotation direction features interesting out-of-equilibrium processes such as vortex - antivortex annihilation and the emergence of other low energy excitations. Furthermore, it has been shown that a three and four vortex-system with one counter-rotating vortex can already lead to chaotic dynamics [54–56] and that large vortex systems can give rise to quantum turbulence [53, 57–59]. Figure 2.1(c)-(d) displays a density and phase profile snapshot during a representative time evolution after phase imprinting additional anti-vortices. While the vortex cores are still clearly visible in the image of the condensate phase, it is more challenging to pinpoint their exact location in the density snapshot.

2.3 Machine learning model

In the following we introduce our neural network based vortex detector which is motivated by state-of-the-art object detectors such as YOLO and Objects as Points [8, 262]. The general task of object detection is to locate each object in an image, draw the corresponding bounding boxes, and associate them to a specific class. Here, we are only interested in detecting vortices and therefore our problem reduces to that of binary classification. In Section 2.5.2 we consider the case where the detector also learns to

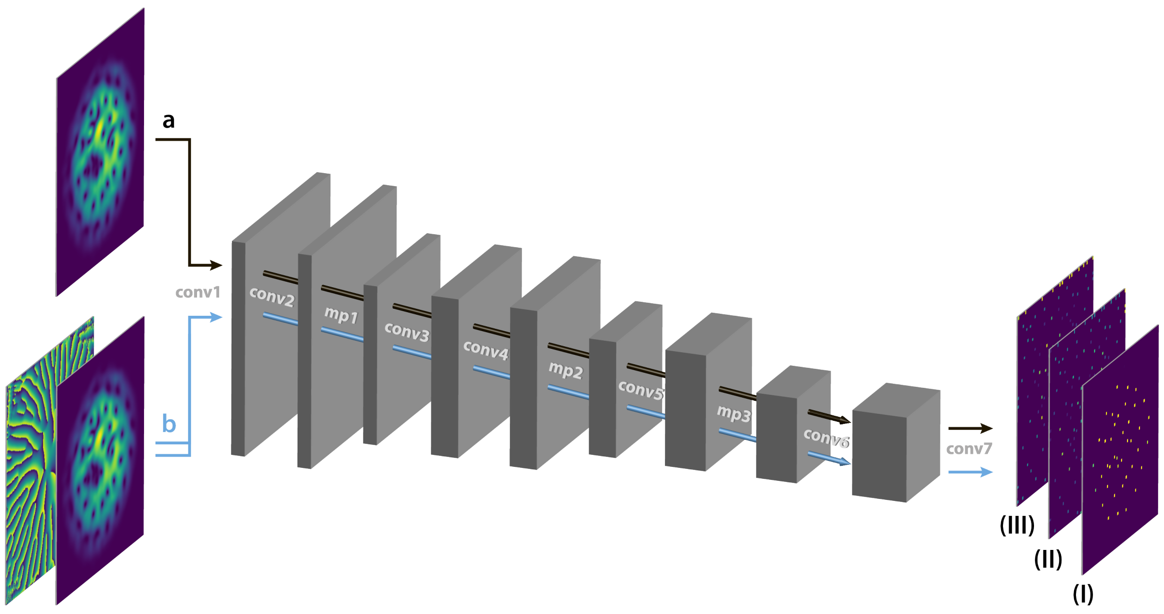


Figure 2.2: The network takes as input images of dimension 256×256 with the condensate density alone (a), or the density and phase profile as two separate channels (b). The images are fed through 7 convolutional and 3 maxpool layers until the final layer outputs 3 matrices of dimension 64×64. Each entry of the 64×64 matrices is associated with a distinct 4×4 cell in the original image and represents the probability of a vortex core being present inside the cell (I), and the scaled x (II), and y (III) position of the vortex within that cell.

distinguish between vortices and anti-vortices as two separate classes. Furthermore, since the sizes of vortices across the simulated images do not vary significantly, we focus on predicting the position of each vortex core rather than the full bounding boxes. If necessary the size of a vortex core can be determined by calculating the healing length of the condensate.

The vortex detector takes as input gray-scale images $I \in [0,1]^{W\times H\times C}$ with equal width and height, $W = H = 256$, and a number of channels $C = 1, 2$ depending on whether the density profile or both density and phase profiles are provided to the neural network in two separate channels (see Fig. 2.2). In principle, the output of the detector can assign a probability to each image pixel corresponding to whether the pixel represents a vortex core or not. However, due to the large dimensions of the input image, we divide it into a $\frac{W}{R} \times \frac{H}{R}$ grid with $R = 4$ such that each 4×4 grid cell is responsible for detecting at most one object. We estimated the size of vortices in our data set and thus, ensured that the grid is chosen fine enough such that at most one vortex is present in any cell. The output Y_{ijk} of the neural network is therefore a tensor of dimensions $64 \times 64 \times 3$ where the 3 channels correspond to the probability of a vortex core being present, and the scaled x and y positions of the core within its grid cell.

In the following, we denote the neural network prediction by Y and the ground-truth label by $\hat{Y}$. The latter are obtained by a brute-force detection method described in detail in Section 2.4.1. Our training and test data is comprised of both ground state and out-of-equilibrium configurations which are obtained through numerical simulations of the GPE (see Eq. (2.1)) with parameter values sampled uniformly from the range $g \in [50, 600]$ for the interaction strength and $\Omega \in [0.65, 0.95]$ for the rotation frequency. The obtained density and phase profiles are normalized such that their pixels lie between $[0, 1]$ before being input to the convolutional neural network (CNN). The architecture is composed of 7 convolutional layers and 3 maxpool operations (see Fig. 2.2). The full details of the architecture, the training, and the chosen hyperparameters are provided in Section 2.4.2. We use the ADAM optimizer [263] and a loss function C given by

$$\begin{aligned} C = \sum_{\text{batch}} \sum_{ij} \Bigg[&- w_1 \hat{Y}_{ij1} \log\left(Y_{ij1}\right) - (1 - \hat{Y}_{ij1}) \log\left(1 - Y_{ij1}\right) \\ &+ w_2 \hat{Y}_{ij1} \left(\left(\hat{Y}_{ij2} - Y_{ij2}\right)^2 + \left(\hat{Y}_{ij3} - Y_{ij3}\right)^2 \right) \Bigg], \end{aligned} \tag{2.2}$$

where w_1, w_2 are hyperparameters. The first term in the loss function is the weighted cross entropy loss responsible for learning the correct assignment of vortex probabilities to each grid cell. We found that giving a higher weight to learning positive predictions stabilizes training since otherwise the network often learned to detect no vortices at all, likely due to the sparsity of vortices within an image. The last term is a mean-squared error (MSE) loss for the x and y positions of a vortex. Note, that only those entries of Y with an existing vortex core contribute to this part of the loss function while all other entries are ignored and in general have arbitrary values. For evaluating and comparing the performance of the object detector we use widely adopted metrics in the field of object detection such as precision, recall, average precision (AP), and the F1 score that we compute on the test data set. For their definitions we refer to Section 2.4.3.

2.4 Training details

The code used to train and evaluate the machine learning model can be found on GitHub [264].

2.4.1 Training data generation

Object detection belongs to the category of supervised learning tasks and therefore requires labeled training data. In order to include a variety of different vortex configurations for training, we use both ground states and non-equilibrium states generated within different parameter regimes. For each training example an interaction strength g and a rotation frequency Ω are uniformly sampled in the range $g \in [50, 600]$ and $\Omega \in [0.65, 0.95]$. The ground state is obtained through imaginary time evolution using the split-step method [265]. To create out-of-equilibrium configurations containing both vortices and anti-vortices, we employ the method of phase imprinting [260] where between $4 - 7$ vortices are placed at random locations and with random circulation directions. We perform a short imaginary time propagation that simulates a small thermal relaxation of the system and a subsequent real-time evolution after which a snapshot of the condensate wave function is saved. The extracted condensate density

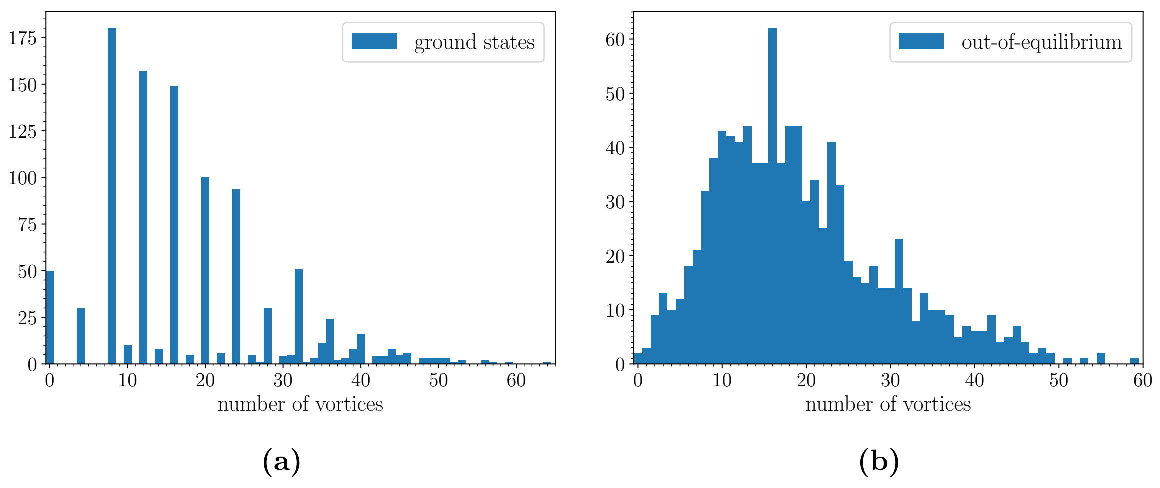


Figure 2.3: Distribution of the number of vortices across the training set for ground states (a) and out-of-equilibrium configurations (b). The combined distribution has a mean value of 18.7 vortices per image.

and phase images comprise the training and test input data which include 1000 ground state samples and another 1000 out-of-equilibrium samples. The combined data set of 2000 images is randomly split into a training set containing 1600 images and a test set including the remaining 400 images.

The ground truth label for each image are the positions of all vortex cores inside the BEC. We obtain the x and y coordinates within pixel resolution through a combination of different techniques. We first apply a mask to the density and phase profiles cutting off regions outside the BEC. Hence, we ensure that only vortices strictly within the condensate are detected. As the cutoff threshold we choose 15% of the maximum density $|\psi|^2$. Next, we find all local density minima within an image. For each local minimum we calculate the phase gradient along a closed loop centered at the minimum, check whether the slope equals ± 1, and the loop adds up to $\pm 2\pi$, thus displaying the characteristic 2π phase winding. If all these conditions are met, the corresponding pixel position is stored in the list of labels. Note that this brute-force method of detecting vortices is not perfectly accurate and misses or mistakenly places vortices in a few cases. Hence, some of the labels used for training are corrupt, however, the overall excellent performance of the detector on the test data suggests that the training of the network is robust against the errors in the training set.

We found that the number of vortices within a single image varied between 0 to 65 in our data set. The distribution of the number of vortices across all images is shown in Fig. 2.3. The dependence of the number of vortices on the applied rotation frequency Ω is nonlinear, i.e., for a large range of sampled rotation frequencies the number of vortices increases only slowly, while in the high frequency regime the number grows more rapidly [266]. The effect can be observed in the distributions which are asymmetric and slightly shifted towards a lower number of vortices with a mean value of 18.7 vortices per image.

Layer	Filter	Stride	Pad	Channels
conv 1	3×3	1	1×1	$1/2^* \to 10$
conv 2	3×3	1	1×1	$10 \to 10$
maxpool 1	2×2	2		$10 \to 10$
conv 3	3×3	1	1×1	$10 \to 20$
conv 4	3×3	1	2×2	$20 \to 20$
maxpool 2	2×2	2		$20 \to 20$
conv 5	3×3	1	1×1	$20 \to 30$
maxpool 3	2×2	1		$30 \to 30$
conv 6	3×3	1	1×1	$30 \to 40$
conv 7	1×1	1		$40 \to 3/4^*$
maxpool 4	3×3	1	1×1	$3/4^* \to 3/4^*$

Table 2.1: Neural network architecture. The number of input and output channels (marked with a star) differ between learning tasks. The final maxpool layer serves as non-max suppression.

2.4.2 Neural network architecture and training

The architecture of the neural-network based vortex detector is based on SlimNet, a convolutional neural network (CNN) specifically designed for detecting small objects [267]. It contains convolutional as well as maxpool layers as depicted in Table 2.1. The network takes as input images of size 256×256 which can be either the density profile or the density and phase profiles provided in two separate channels. All convolutional layers are followed by relu activations except for the last layer, which uses a sigmoid activation instead. The output Y of the network after the final convolutional layer is a $64 \times 64 \times 3$ tensor where each of the three channels corresponds to the probability of detection, and the scaled x, and y positions respectively. For example, an output $Y_{ij1} = 1$ would indicate that a vortex is present in the grid cell denoted by ij and the precise position of the core within that grid cell can be read off by checking Y_{ij2} for the x and Y_{ij3} for the y coordinates. On the other hand, all grid cells where $Y_{ij1} = 0$ do not contain a vortex and therefore the second and third output channels can be ignored. In the case where the circulation direction of a given vortex is also classified, the output contains 4 channels with the first two representing the probability for a vortex and anti-vortex respectively. The final maxpool layer serves as a non-max suppression to eliminate multiple detections of the same vortex.

We implement the CNN and train it using Julia's machine learning library Flux [268]. We use the ADAM optimizer [263] with a batch size of 100, a learning rate $\eta = 0.001$, and decay rates $\beta_1 = 0.9$ for the first and $\beta_1 = 0.999$ for the second momentum estimates. The weights in the loss function of Eq. (2.2) are set to $w_1 = w_2 = 10$ and the network is trained for $100 \sim 500$ epochs depending on the learning task. 100 epochs of training took on the order of 10 minutes on a NVIDIA TITAN X Pascal GPU. A subsequent forward pass took 0.0025 sec for a single image and 0.009 sec for a batch of 100 images therefore allowing for real-time detection once the model is successfully trained.

2.4.3 Evaluation metrics

To evaluate and compare the performance of the vortex detector for each different learning task we use standard metrics in the field of object detection such as precision and recall [241]. Precision describes how many of the detections within an image are accurate, while recall quantifies how many of the actual objects in an image are detected. Denoting true positives by TP, false positives by FP, and false negatives by FN, precision and recall are defined through

$$\text{Precision} = \frac{TP}{TP + FP} \,, \tag{2.3}$$

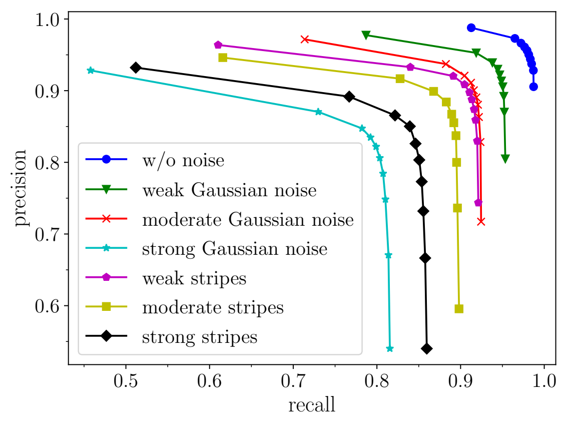
$$\text{Recall} = \frac{TP}{TP + FN} \,. \tag{2.4}$$

The machine learning model outputs the probability for a vortex to be present in each grid cell. A confidence threshold is used to discard low probability detections and label high confidence predictions as positives P. Conventionally, a larger confidence threshold increases precision while decreasing recall and vice versa. For each different detection task we calculate an optimal confidence threshold that maximizes the harmonic mean of precision and recall, which we describe further below. To distinguish true positives from false positives, the object detection community usually computes the intersection over union (IoU) given by the ratio of the intersection area and union area of the detected bounding box and the ground-truth bounding box. If the IoU is larger than a predefined value, the detection is considered to be a true positive TP while it is labeled as a false positive FP otherwise. In our case the vortices across images are of similar sizes and therefore we can use a simple distance measure between the detected vortex position and the ground-truth position instead of the IoU. We choose the pixel-wise euclidean distance and an arbitrary distance thresholds of $\sqrt{5}$. Hence, all detections that are within ~ 2 pixels of their ground-truth position are identified as positives.

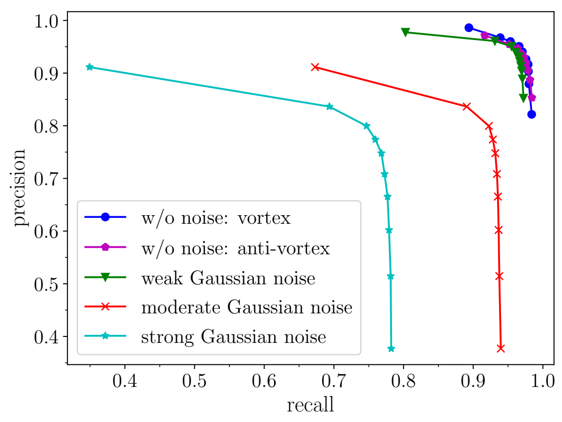
As mentioned before there exists a trade-off between precision and recall controlled by the value of the confidence score threshold. To examine the performance of the model across different confidence thresholds, we plot precision against recall for 10 different threshold values in Fig. 2.4. Each curve of a different color corresponds to one of the separately trained models. From each point on a curve we can determine the F1 score, i.e., the harmonic mean between precision and recall

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \,. \tag{2.5}$$

The optimal confidence threshold corresponds to the maximum F1 score which is used for generating all labeled plots within this work. Furthermore, we calculate the average precision (AP) as the mean over the precision values $p(r)$. Finally, we also provide the precision and recall values computed at the optimal confidence threshold as another meaningful performance metric of the vortex detector. In the case of multi-class detection considered in Section 2.5.2 the AP and F1 scores are first calculated separately for each class and then averaged to obtain mean average precision and a mean F1 score.



(a)



(b)

Figure 2.4: Precision - recall curves for the training tasks of Section 2.5.1 (a) and Section 2.5.2 (b). Precision/recall values are calculated for 10 different confidence thresholds in the equally spaced interval between [0.05, 0.95]. The precision/recall values corresponding to the highest F1 score are shown in Table 2.2 together with the maximum F1 score and the average of the precision values (AP).

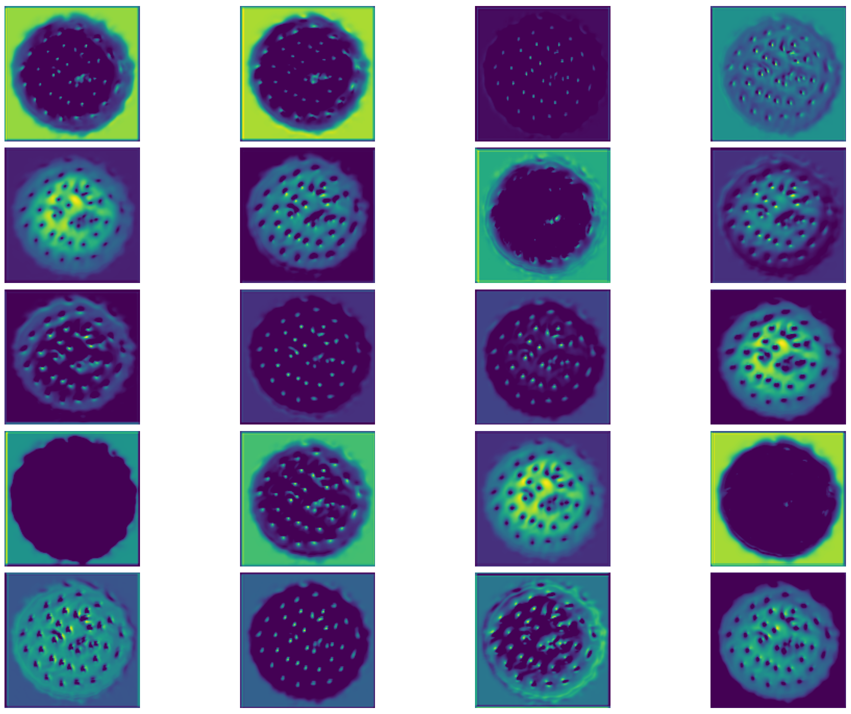
2.4.4 Visualization of the CNN layers

To elucidate the inner workings of the trained CNN we visualize the output after the fourth convolutional layer for a particular input image in Fig. 2.5. Each image corresponds to the output of one of the 20 channels and gives information about the features learned by the network. Figure 2.5(a) shows the feature maps after training on only BEC density images. Here, the network clearly learns to separate the condensate from the background by applying different masks. It also detects all density minima within the condensate, however, each channel seems to focus on slightly different characteristics such as the depth, size, or shape of a minima. On the other hand, the feature maps plotted in Fig. 2.5(b) were obtained when training with phase and density images. In this case, an interpretation is less evident, but we can observe that the network takes advantage of the additional information supplied by the phase profile. Interestingly, we found that the output of CNN layers trained on noisy images did not differ significantly from the ones shown here. Therefore, the model also learns to de-noise the input if necessary and thus, ignores any spurious features contained in the noise itself.

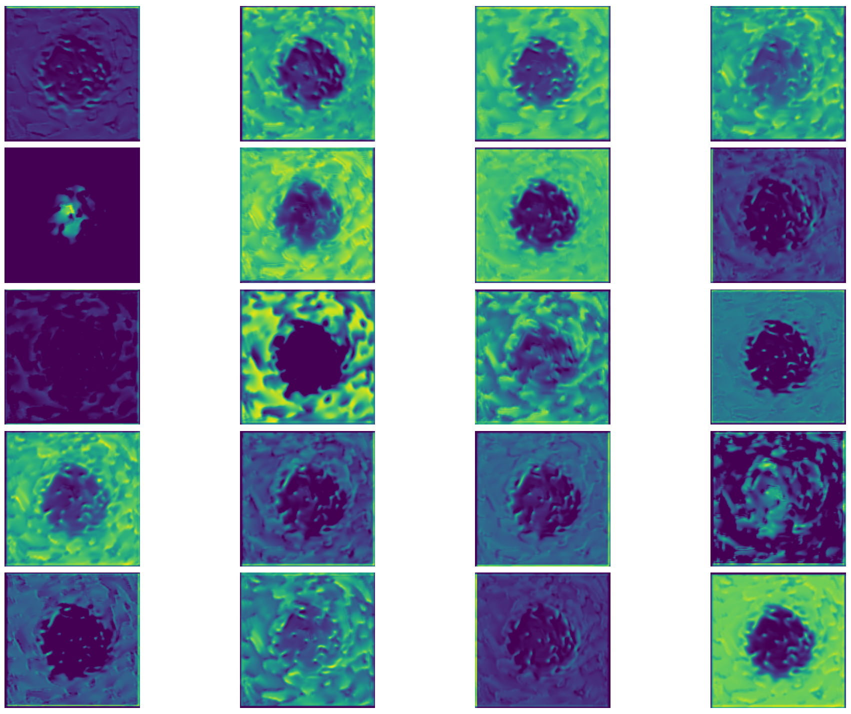
2.5 Results

2.5.1 Vortex detection using density only

First, we train the object detector directly on density images obtained from simulations with the GPE, i.e., without any addition of noise. Figures 2.6(a)-(b) show two representative density images with white circles corresponding to the ground truth and red crosses to the prediction of the trained model. Overall, we achieve a precision of 96.6% and a recall value of 97.2% on the test data (all other computed evaluation metrics can



(a)



(b)

Figure 2.5: Example output after the 4th convolutional layer of the CNN when training only on density images (a) or on density and phase images (b).

be inferred from Table 2.2). Precision and recall are calculated through comparison with the ground truth position obtained from the brute-force detection method which is not always accurate itself. Hence, our CNN likely performs better than the computed metrics.

While the network detects all vortices in Fig. 2.6(a) with nearly perfect accuracy, we observe deviations from the ground truth label in the example shown in Fig. 2.6(b). Here, the model detects additional vortices at the boundary of the condensate. However, the corresponding phase profile in Fig. 2.6(c) features the characteristic phase winding at the location of the additional detections and hence these can be interpreted as vortices as well. In general we found that in most of the cases where the number of ground-truth detections and model detections differ, the missing/additional vortices lie at the boundary of the BEC and are often accompanied by a lower confidence probability. Note that the ground-truth labels were obtained using a brute-force detection algorithm which involves applying an arbitrarily chosen mask to the density images cutting off low density regions and therefore excluding any vortices that are not strictly within the BEC (see Section 2.4.1 for further details). Hence, during training the neural network also learns that vortices located in very low density regions should not be detected as such, however, it does not have access to the specific mask used in the brute-force detection algorithm. Therefore, it likely learns a slightly different density cutoff which gives rise to the additional detections in the test data.

To emulate experimental conditions we trained separate networks on images with two different sources of noise. The first type is Gaussian random noise with mean zero which is added to each pixel of the normalized condensate density images and mimics the measured density distributions in for example Ref. [269]. We trained three independent CNNs each with a different level of noise, i.e., a different standard deviation ($\sigma = 0.1, 0.2, 0.5$), and plot the resulting predictions together with their ground-truth in Fig. 2.6(d)-(f). As a another example of experimentally relevant noise we consider stripes in the density images which resemble the fringe patterns that can arise in absorption imaging due to unwanted interference effects [270, 271]. To mimic this pattern we add a sinusoidal modulation to the density with randomly chosen direction and period. In addition, we add Gaussian random noise to the amplitude of the modulation itself. Figures 2.6(g)-(i) show the corresponding density images together with the model prediction where the amplitude A of the modulation and the amount of noise increases from left to right ($A = 0.2, 0.5, 1.0,\ \sigma = 0.2, 0.5, 1.0$). As expected, for both considered types of noise the performance of the detector deteriorates as the amount of noise increases which is also reflected by a smaller precision and recall value (see Table 2.2). In general we found that, as the noise grows, first only the predicted vortex positions become less accurate while for larger amounts of noise the model starts to entirely miss or mistakenly place vortices.

Note that although we have trained separate models for each different level of noise, we observed that each of the trained networks is able to generalize well to a different strength and type of noise which is crucial for real experimental situations where the amount of noise will likely change between measured images and experimental runs. For example, the model trained solely on strong Gaussian noise with $\sigma = 0.5$ achieved both a precision and recall of approximately 90% on the test images containing a lower amount of noise and hence performs only slightly worse than the networks that have

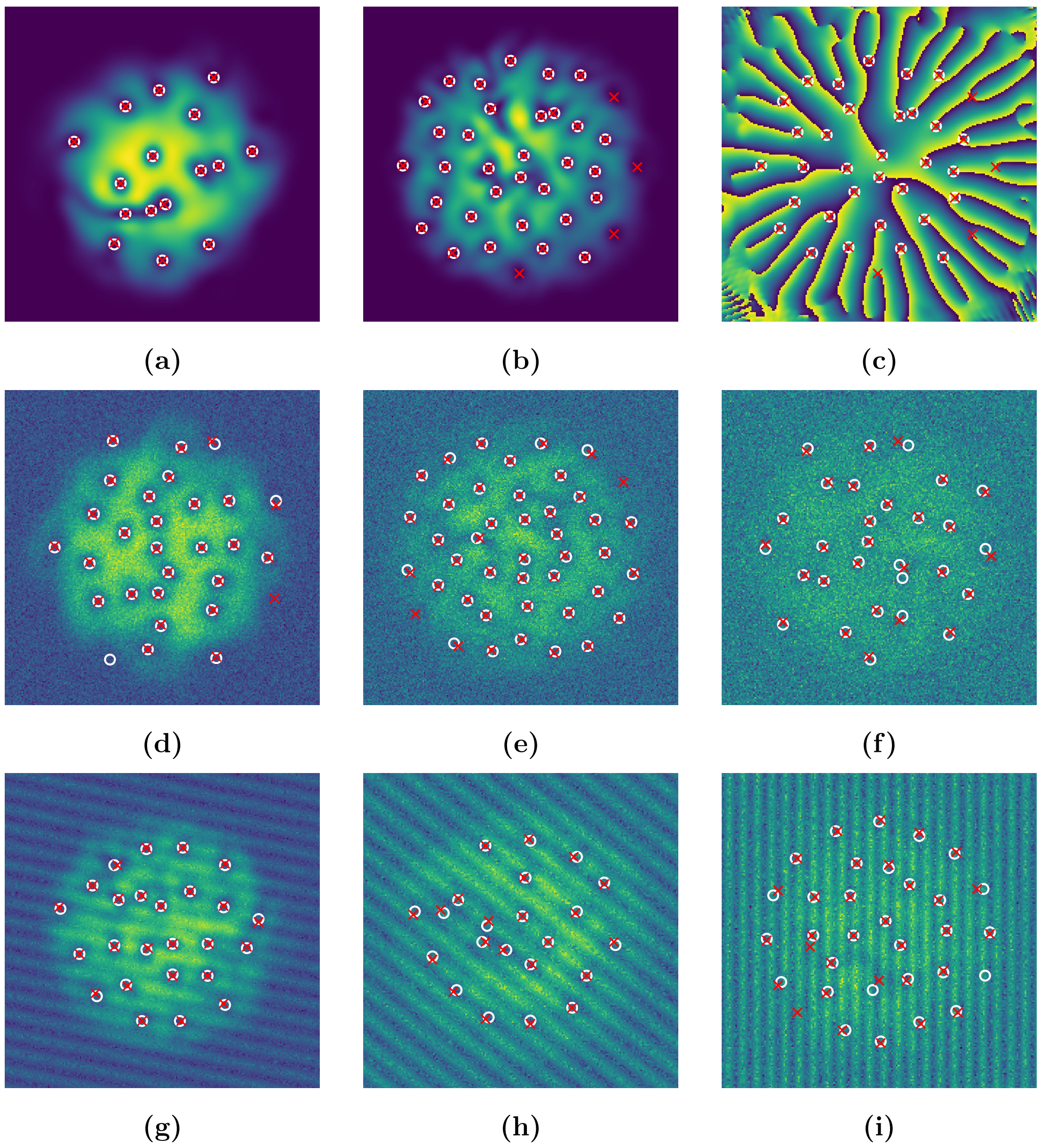


Figure 2.6: The locations of the vortex cores within each image are indicated by red crosses for the model prediction and by white circles for the ground truth obtained through the brute-force detection method. The CNN model was trained and tested only on BEC density images and therefore does not have access to the information encoded in the phase profile. (a) and (b) show two examples of BEC density configurations while (c) is the corresponding phase profile for the density image in (b) provided here as a guide for the eye. (d) - (f) display density distributions to which random Gaussian noise is added with growing standard deviations from left to right ($\sigma = 0.1, 0.2, 0.5$). In (g) - (i) a sinusoidal modulation with Gaussian noise is added instead where the amplitude A and the amount of noise increase from left to right ($A = 0.2, 0.5, 1.0$). Note that the pixels in the density images are normalized to lie between $[0, 1]$ before including any noise and before being fed to the neural network.

	Precision	Recall	AP	F1
Detection using density only (fig. 2.6)				
(a),(b) w/o noise	96.6	97.2	95.1	96.9
(d) weak Gaussian noise	93.9	93.8	91.1	93.9
(e) moderate Gaussian noise	92.1	90.5	88.2	91.3
(f) strong Gaussian noise	84.7	78.2	78.5	81.3
(g) weak stripes	90.9	90.5	88.2	90.7
(h) moderate stripes	88.4	88.3	83.4	88.4
(i) strong stripes	85.0	83.9	78.8	84.5
Detection using density and phase (fig. 2.7)				
(a),(d) weak Gaussian noise	95.1	95.5	92.4	95.3
(b),(e) moderate Gaussian noise	92.4	92.3	88.0	92.3
(c),(f) strong Gaussian noise	78.0	74.7	69.4	77.2
Detection and classification (fig. 2.8)				
w/o noise, vortex/anti-vortex	94.9	96.5	92.3	95.7

Table 2.2: Detector performance metrics (precision, recall, (mean) average precision (AP), and maximum F1 score) computed on the test data for each trained model (see Section 2.4.3 for their definitions). In the case of detection using BEC density images only, the Gaussian noise is added to the normalized density distributions with mean zero and standard deviations $\sigma = 0.1$ (weak), $\sigma = 0.2$ (moderate) $\sigma = 0.5$ (strong). The stripe pattern was achieved by adding a sinusoidal modulation instead with amplitudes $A = 0.2$ (weak), $A = 0.5$ (moderate), $A = 1.0$ (strong). Finally, in the case of using both the density and the phase profiles as input to the CNN, the Gaussian noise was added directly to the real and imaginary parts of the wave function.

been directly trained on those data sets. The same model also performed well on images with weak stripes, however, for the case of moderate and strong stripes the model performance is considerably worse. This trend is however expected since the stripe pattern contains unique features that the model has not been exposed to during training. We also found that a network trained on a lower noise level generalizes to a certain extent to data involving more noise. For instance, the network trained on images with weak stripes achieves good accuracies on the test images with weak/moderate Gaussian and stripe noise with $F1$ scores over 80%, and only has difficulties locating vortices in images with a very large amount of noise. A summary of the computed evaluation metrics for the two examples discussed here is given in Table 2.3 in Section 2.5.3.

2.5.2 Vortex detection using density and phase

In numerical simulations of the GPE one has access to the full mean-field condensate wave function rather than just its density. Hence, we can provide both the density and the phase profile as input to the CNN in two separate channels in analogy to the three color RGB channels of a conventional image. In order to make the detection more challenging, we add Gaussian random noise to the real and imaginary part of the wave

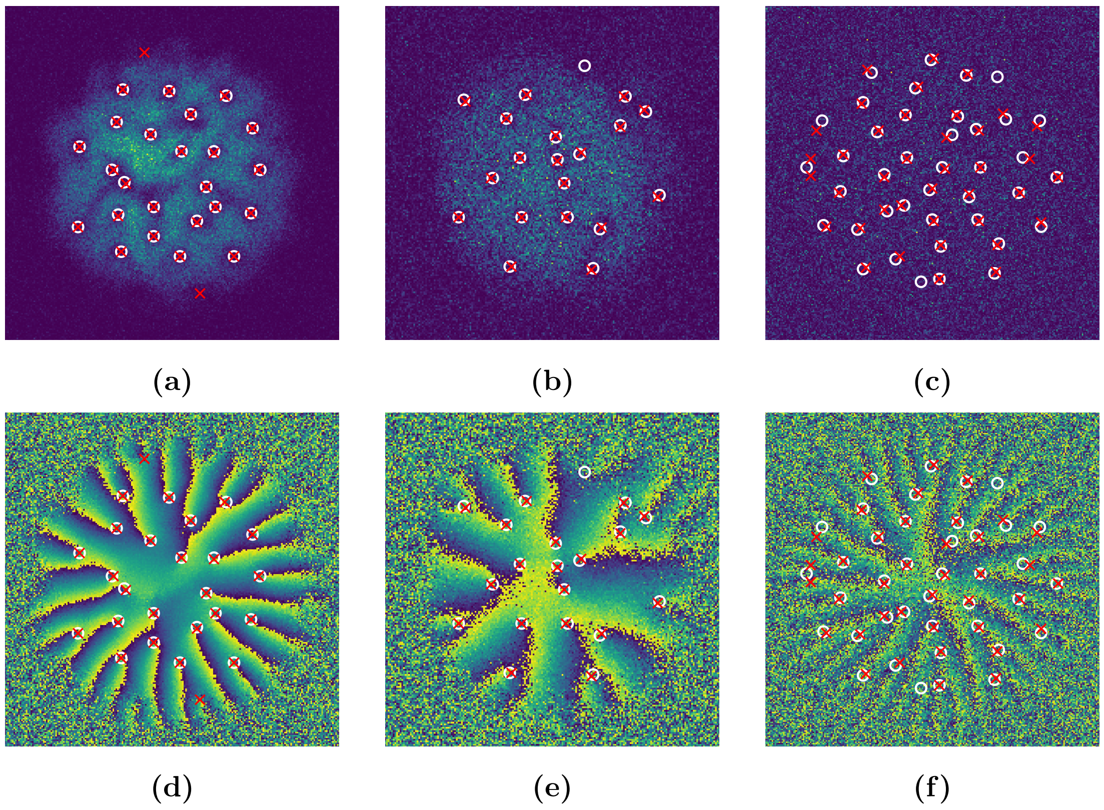


Figure 2.7: BEC density (a)-(c) and corresponding phase (d)-(f) configurations after adding Gaussian random noise to the real and imaginary parts of the condensate wave function. The amount of added noise increases from left to right. The model prediction is again denoted by red crosses while the ground truth is indicated by white circles. All images are normalized to lie between [0, 1].

function which gives rise to the density and phase distributions depicted in Fig. 2.7(a)-(c) and Fig. 2.7(d)-(f) respectively. We train three models on different levels of noise and show the predicted vortex locations together with their ground truth in Fig. 2.7. The achieved performance metrics can be read off Table 2.2. Especially for the case of weak noise, the detector performs very well with precision and recall values both above 95% suggesting that the network exploits the additional information encoded in the BEC phase profile.

Furthermore, having access to the phase profile allows us to determine the direction of circulation through the sign of the phase winding. In the images shown in this chapter (see for example Fig. 2.8(d)-(f)) the circulation direction can be easily inferred by checking the direction of the color gradient when moving in a clock-wise loop around a vortex core, i.e., whether the color changes continuously from yellow to blue or vice versa. Hence, we next train the network to also classify the sign of circulation for each vortex. The CNN is slightly altered to output 4 channels, the first two now corresponding to the probability of vortices and anti-vortices in a specific grid cell and the last two channels again contain the information about the precise location of a detected vortex. The loss function in Eq. (2.2) is changed accordingly. Figure 2.8 shows three exemplary density and phase images where the model prediction is

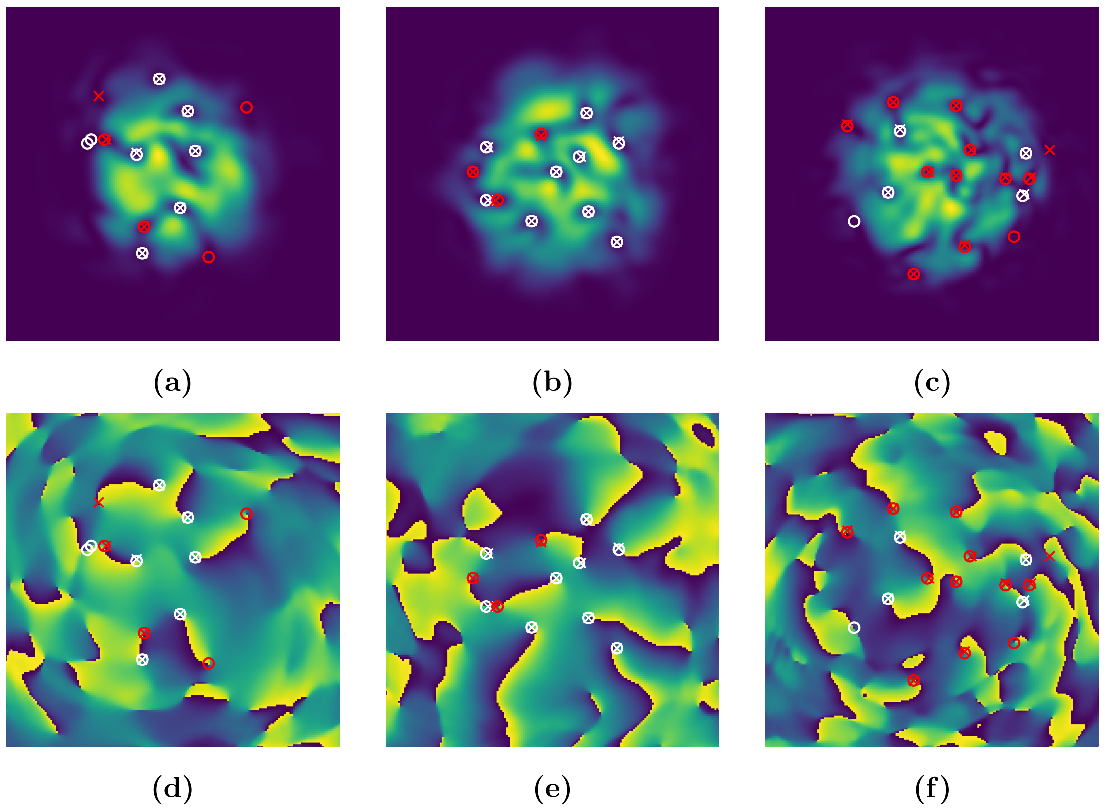


Figure 2.8: BEC density (a)-(c) and corresponding phase (d)-(f) configurations. The predicted vortex locations from the network are indicated by crosses and the ground truth by circles. The model was trained to also classify the circulation direction of a vortex with vortices depicted in red and anti-vortices in white.

represented as crosses and the ground-truth as circles while vortices are depicted in red and anti-vortices in white. The model is able to accurately distinguish the circulation direction and in particular classifies all windings correctly for the images shown here. Moreover, it finds all vortices within the high-density region of the condensate which is also reflected by high precision and recall values as shown in Table 2.2.

2.5.3 Generalization to different sources and levels of noise

The models considered above were all trained on data sets containing a specific source and level of noise. However, real experimental images will not be subject to just a single source of noise and the amount of noise will likely vary between images. Therefore, we also test how well a model trained on a specific noise configuration generalizes to other types and levels of noise. As two examples we consider a model trained only on data with strong Gaussian noise or only on images with weak stripes. We show the computed performance metrics for both models tested on all different data sets in Table 2.3. The values suggest that the networks indeed generalize to unseen noise strengths and types especially if the amount of noise the model is tested on is lower compared to the one present in the training data.

	Precision	Recall	AP	F1
(a) Model trained on data with strong Gaussian noise and tested on data with:				
w/o noise	91.2	91.4	82.1	91.3
weak Gaussian noise	92.3	89.3	83.3	90.8
moderate Gaussian noise	91.0	87.7	83.3	89.3
strong Gaussian noise*	84.7	78.2	78.5	81.3
weak stripes	87.6	81.3	71.9	84.3
moderate stripes	65.9	58.4	51.0	61.9
strong stripes	46.4	44.8	37.8	45.6
(b) Model trained on data with weak stripes and tested on data with:				
w/o noise	91.6	91.6	71.1	91.6
weak Gaussian noise	90.1	82.8	86.6	86.3
moderate Gaussian noise	83.1	76.1	85.8	79.5
strong Gaussian noise	61.6	41.0	64.2	49.3
weak stripes*	90.9	90.5	88.2	90.7
moderate stripes	83.4	77.1	79.4	80.1
strong stripes	61.7	53.1	58.3	57.1

Table 2.3: Detector performance metrics (precision, recall, average precision (AP), and maximum F1 score) for a model trained on images with (a) strong Gaussian noise ($\sigma = 0.5$) and (b) weak stripe noise ($A = 0.1$). Each of the two models is tested on all data sets containing a different strength and/or source of noise. The type of noise marked with a star represents the corresponding data set on which the model was trained. The Gaussian noise is added to the normalized BEC density with mean zero and standard deviations $\sigma = 0.1$ (weak), $\sigma = 0.2$ (moderate) $\sigma = 0.5$ (strong). The stripe pattern resulted from adding a sinusoidal modulation to the normalized density with amplitudes $A = 0.2$ (weak), $A = 0.5$ (moderate), $A = 1.0$ (strong).

2.5.4 Generalization to different trap geometry

The networks considered in this work have been trained solely on BEC configurations in a harmonic trap. To examine whether the same model can be used to detect vortices in different trap geometries, we generate additional images of a BEC in a ring shaped trap. The potential in the GPE (Eq. (2.1)) is replaced by $V = \frac{1}{2}m\omega_r^2(r - r_0)^2$ with r_0 being the toroidal radius and ω_r the radial trapping frequency. An example of a resulting condensate density and phase profile after real time evolution is shown in Fig. 2.9(a),(d). We test some of our previously trained networks on these new configurations without any further training and indicate the predictions together with their ground truth in Fig. 2.9. The model is able to accurately locate the vortices within the high density regions for images without (Fig. 2.9(a)) and with noise (Fig. 2.9(b)-(c)) while it detects additional vortices at the border of the condensate. Increasing the confidence threshold will likely result in fewer false positive detections. Due to the overall good performance we observe for the ring shaped potential, we expect that our trained models generalize

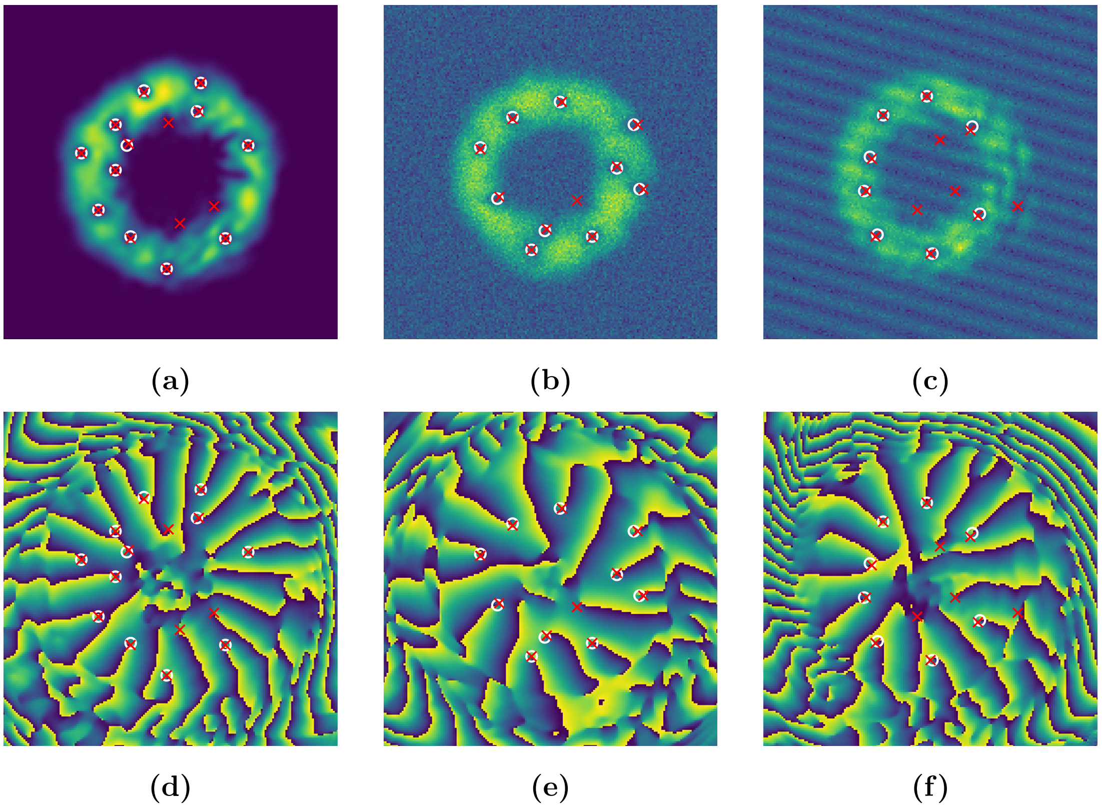


Figure 2.9: Density (a)-(c) and corresponding phase (d)-(f) configurations for a BEC in a ring shaped trap. A weak Gaussian noise is added to the density in (b) and a stripe pattern is added to the density in (c). Each density image is fed through the corresponding trained network with the resulting predictions displayed as red crosses and the ground truth as white circles. The phase profile is shown for pure visualization purposes that is, only the density images were used as input to the CNN.

well to other trap geometries of similar symmetry.

2.6 Conclusion

In this work we have presented a machine learning based vortex detector that can accurately predict the locations of vortices within two-dimensional BECs trapped in harmonic potentials. The machine learning model is based on a convolutional neural network (CNN) and takes as input either an image of the BEC density only or both, the BEC density and phase profiles. We first studied the experimentally more relevant case where only the condensate density is available and thus used for training and testing. Without any sources of noise the detector is able to reliably locate all vortices within an image. Moreover, the model performs well on non-equilibrium configurations that involve local density minima not corresponding to vortices. Hence, it learns to distinguish density minima arising due to vortices from those caused by other low energy excitations even in cases that are challenging for the human eye.

To simulate more realistic experimental conditions, we trained the network on density images with two different types of added noise that is, Gaussian noise and spurious stripe patterns arising due to unwanted optical interference effects [269–271]. In either case, the achieved accuracy of the trained detector decreases with the amount of added noise as expected. However, overall the performance is still impressive given that it is nearly impossible to locate any vortices by eye in those images that contain a high level of noise. In contrast to the experimental setting, in numerical simulations both the density and phase profile of a BEC are available and therefore can be used to also distinguish the circulation direction of each vortex. In this case our detector learns to correctly classify the sign of circulation as well. Finally, the network is also able to accurately locate vortices in noisy configurations where Gaussian noise is added directly to the wave function itself. Due to the robustness of the detection against noise, it might be promising to train the detector on configurations generated by the stochastic GPE, which models BECs at finite temperatures [237, 272, 273].

We trained and tested the CNN using ground-truth labels obtained through a brute-force detection algorithm which already provides the position of vortices to a very good precision. This raises the question whether a machine learning approach is even necessary and advantageous. However, the brute-force detection algorithm has certain disadvantages: It involves searching for density minima and checking whether the conditions for a vortex, such as a 2π phase winding, are fulfilled which is neither efficient nor easily parallelizable. Furthermore, the algorithm does not achieve perfect accuracy itself, i.e., it misses vortices or mistakenly places them at times. Additionally, the method heavily relies on the phase profile of the BEC for labeling out-of-equilibrium configurations, where local density minima may arise due to other excitations in the quantum system. However, in experiments the phase information is not easily accessible and therefore the algorithm cannot be straightforwardly applied in these settings. Finally, our brute-force method only works for simulated images without noise. While it is in general possible to improve the algorithm to also detect vortices in images with specific sources of noise, the implementation would be considerably more cumbersome. On the other hand, our machine learning based detector is robust to various sources of noise in the input data and does not rely on any hand engineered features.

The presented network can be trained in less than an hour on a single GPU and did not require elaborate hyperparameter tuning for any of the tasks considered here. Furthermore, the CNN is able to process several images in parallel and can therefore detect vortices in a large batch of input images fast given that the computation is performed on a GPU. For example, processing a batch of 100 images takes only on the order of milliseconds. The machine learning model can also be straightforwardly integrated with a GPU solver of the Gross–Pitaevskii equation which would eliminate the need to transfer data between CPU and GPU [274]. As a possible next step the vortex detector could be combined with a tracking algorithm enabling the study of real-time dynamics of vortices in BECs such as in Refs. [54–56]. It would also be interesting to compare the performance of our model to different architecture choices and object detection techniques, and we hope that our results can serve as a benchmark for further research on quantum vortex detection methods.

While the neural network has only been trained on images of BECs in a uniform harmonic trap, we found that the same model can detect vortices in ring-shaped traps without any additional training, and hence we expect that the detector also generalizes to other trapping geometries of similar symmetry. Moreover, we observed that a model trained on a particular strength and type of noise also worked well on different levels of noise. The promising generalization capabilities and the fact that the model performs well on density images alone with sources of noise and in the presence of spurious density minima suggests that the detector will be advantageous for experiments studying the dynamics of vortices in non-equilibrium states [57, 231, 232].

Our work was one of the first that applied deep-learning based object detection techniques to AMO physics with the goal of automating experimental data postprocessing tasks. Shortly after our paper a similar work by Guo et al. appeared [246]. They employ a convolutional neural network for detecting and locating single solitons in experimental density images of BECs. Guo et al. [275] later generalized this framework to the detection of multiple solitons per image by borrowing techniques established in our work. Moreover, our vortex detector has been used by Kim et al. [276] for locating vortices in experimental BEC density images of ^{87}Rb atoms trapped in an external potential. The vortex detector allowed Kim et al. to study the spontaneous formation of vortices under quench dynamics of the Bose gas. This work therefore demonstrates that our framework can also be successfully applied to experimental data.

Chapter 3

Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer

The text in this chapter is largely based on the following publication, but extended to provide additional content.

Korbinian Kottmann[*], **Friederike Metz**[*], Joana Fraxanet, and Niccolò Baldelli
Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer
Phys. Rev. Research **3**, 043184 (2021)

This project developed as part of a quantum computing hackathon that me and my co-authors participated in. I implemented the quantum autoencoder in Qiskit (Python) and worked on the VQAD simulations for the TLFI and DEBHM model. All authors contributed to the discussions, the interpretation of the results, and the writing of the manuscript. Parts of this chapter are also contained in the PhD thesis of Korbinian Kottmann [277], but each of us only submit our specific contributions for the evaluation for our respective PhDs.

3.1 Introduction

Quantum simulation is one of the most promising applications of near-and far-term quantum computing [49]. Current noisy-intermediate scale quantum (NISQ) devices still feature too large error rates and too few qubits to allow for quantum error correction and consequently fault-tolerance. However, the development of variational quantum algorithms and error mitigation techniques already opens the door to small-scale quantum simulations of condensed matter systems and molecules [45–48]. Notable examples of these hybrid quantum-classical algorithms are the variational quantum

*indicates co-first authorship

eigensolver (VQE) [278] or the quantum approximate optimization algorithm (QAOA) [279] for ground state search, and the time-dependent variational algorithm (TDVA) [280] or projected-variational quantum dynamics (p-VQD) [281] for time evolution. Yet, with the rapid progress in quantum simulation techniques also grows the need for tools to analyze the simulated quantum systems natively on the respective devices.

Meanwhile, the field of classical computational many-body physics has markedly benefited from harnessing machine learning techniques to study the properties of quantum systems [10, 81]. A prominent example is the use of deep neural networks for classifying phases of matter [13, 14]. These schemes rely on labeled training data and hence, often require some prior knowledge about the system at hand such as the number of different phases and their approximate locations in the phase diagram. However, recently fully unsupervised techniques have been proposed as well which eliminate the need for expert knowledge or prior intuition altogether [18–20, 64, 186, 187].

On the other hand, the analogous field of quantum machine learning (QML) is relatively young. In QML parameterized quantum circuits are trained to solve typical learning problems like classification or generative modeling [47, 50–52]. However, quantum machine learning applications to quantum states and in particular to the problem of quantum phase classification are still limited.

In this work, we propose a variational quantum algorithm based on the idea of anomaly detection for investigating quantum systems in a fully automated and unsupervised fashion. In anomaly detection one generally tries to find anomalous examples in an otherwise homogeneous data set of “normal” data points. The model output that determines to which type the input data belongs is referred to as an anomaly syndrome. The model is trained solely on normal data points for which it outputs a specific value (typically zero). Hence, it does not require any training labels. When confronted with unseen anomalous data during testing the anomaly syndrome results in a different value, thus indicating that the input features different properties than the training data.

Our algorithm can be considered as the quantum analog of the classical anomaly detection framework introduced in Ref. [18]. Here, the authors trained a neural network-based autoencoder to reproduce ground states in one of the phases of the extended Bose Hubbard model. The cost function served as the anomaly syndrome and by plotting its value across ground states for different parameter regimes, they were able to recover the full phase diagram. This method was later applied to topological, frustrated, and higher dimensional systems [62, 63, 188].

Similarly, we use a quantum autoencoder to perform anomaly detection on ground states that are simulated on a quantum computer. Hence, quantum systems can be studied end-to-end on the same device they are simulated on without requiring any prior knowledge about their phases. We showcase our algorithm by mapping out the phase diagrams of the dimerized extended Bose Hubbard model and the Ising model with transverse and longitudinal fields. Furthermore, we perform our framework in the presence of noise, both on a classical simulator and a real physical quantum device.

This chapter is organized as follows: Section 3.2 introduces the quantum anomaly detection framework and the underlying quantum autoencoder. In Section 3.3, I discuss the results of training the VQAD framework on ground states of the transverse longitudinal field Ising model and the dimerized extended Bose Hubbard model. Finally,

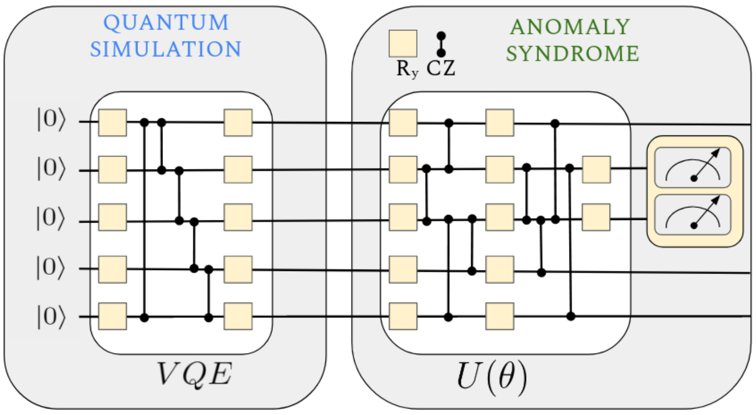


Figure 3.1: Overview of our proposal. First, the quantum states are prepared via VQE. Then, they are processed through the anomaly syndrome, consisting of a parameterized unitary $U(\theta)$ and a measurement of a subset of qubits, referred to as trash qubits. R_y indicates a parameterized y-axis rotation and CZ a (fixed) controlled-z gate.

Section 3.4 provides some additional details regarding the VQAD training.

3.2 Proposal

The task of detecting anomalies in ground states of quantum many-body Hamiltonians can be loosely divided into two sub tasks: Preparing the ground state for specific Hamiltonian parameters, and computing an anomaly syndrome indicating whether the state corresponds to a *normal* example or an anomaly. An overview of our proposed algorithm is shown in Fig. 3.1. The problem of state preparation on quantum computers is one of ongoing research, and in principle, one can use any state preparation subroutine for preparing the ground state. Here, we choose the Variational Quantum Eigensolver (VQE) as it has the lowest hardware requirements while achieving reliable results on current devices [167, 278]. The VQE algorithm iteratively minimizes the expectation value of a Hamiltonian with the ansatz circuit to find the ground state by optimizing the parameters of the circuit via a quantum-classical feedback loop. We choose a minimal ansatz as depicted in Fig. 3.1 that is sufficient for simulating the Ising Hamiltonian discussed in Section 3.3.2. A shallow ansatz allows us to run both, the quantum simulation, and the quantum anomaly detection on real noisy devices. For more complex systems, the problem of finding a suitable hardware efficient ansatz can be addressed for example by the adaptive VQE algorithm [282]. In this work we employed the VQE implementation provided by the Qiskit library [283] and optimized it using simultaneous perturbation stochastic approximation (SPSA) [284]. For all technical details we refer to Section 3.4.

Once the ground state is prepared on the quantum device a subsequent circuit serves as the anomaly syndrome. Our circuit ansatz is inspired by the recently proposed quantum auto-encoder, which similar to its classical counterpart can be used for compression of classical and quantum data [177, 178]. It is composed of several layers each consisting of parameterized single qubit y-rotations and controlled-z gates. After the final layer a predefined number n_t of *trash* qubits is measured in the computational basis. The objective is to decouple the trash qubits from the rest of the system, effectively compressing the original ground state into a smaller number of qubits. The circuit parameters are then optimized to faithfully compress states that are considered *normal*. However, when the optimized circuit is tested on anomalous states not seen during training, it is expected that the circuit fails to decouple the trash qubits from the rest of the system. To quantify the degree of decoupling we use the Hamming distance d_H of the trash qubit measurement outcomes to the $|0\rangle^{\otimes n_t}$ state, i.e., the number of 1s in a bit-string of measurement outcomes [178]. The cost function C can then be defined as the Hamming distance averaged over several circuit evaluations $C = 1/N \sum_i^N d_{Hi}$, where N is the number of performed measurements or shots. The cost function can also be rewritten in terms of expectation values of local Pauli-z operators Z_j

$$C = \frac{1}{N}\sum_{i=1}^{N} d_{Hi} = \frac{1}{2}\sum_{j=1}^{n_t}\left(1 - \langle Z_j\rangle\right). \tag{3.1}$$

The VQAD circuit achieves perfect compression if the trash qubits are fully disentangled from the remaining qubits and mapped into the pure $|0\rangle^{\otimes n_t}$ state resulting in a cost equal to zero.

The specific circuit ansatz for the anomaly syndrome is shown in Fig. 3.1 for the case of $n_t = 2$ trash qubits. Each layer of the circuit starts with parameterized single-qubit y-rotations applied to every qubit followed by a sequence of entangling controlled-z gates. The currently available NISQ devices are inherently noisy and the computations are subject to gate errors. To minimize the number of two-qubit gates we apply the controlled-z gates only between trash qubits and non-trash-qubits as well as between trash qubits themselves instead of an all-to-all entangling map [178]. This entangling map is physically motivated as the goal of the circuit is to disentangle the trash qubits from the rest, with the trash qubits resulting in the $|0\rangle^{\otimes n_t}$ state. In a single layer each non-trash qubit will be coupled to exactly one trash qubit. This entangling scheme is repeated in the subsequent layers until every non-trash qubit has been coupled to each trash qubit exactly once, i.e., the number of layers of the circuit is equal to n_t. After the final layer, additional single-qubit y-rotations act on the trash qubits before they are measured.

Barren Plateaus are the fundamental obstacle prohibiting training of variational circuits with increasing numbers of qubits [173]. It was previously shown that using local cost functions and circuits featuring a number of layers scaling at most logarithmically in the system size can prevent the occurrence of Barren Plateaus [174]. Additionally for realistic devices, gate errors lead to decoherence, making quantum simulation on real devices a challenging task even for small systems and low depths [285]. The former calls for a minimal number of layers while the latter calls for a minimal number of gates overall. Therefore, we seek a minimal solution for our variational circuit that we want to implement on a readily available NISQ-era quantum computer. On the other hand, it is desirable to have an ansatz as general as possible to be able to capture a wide range of problems (see *circuit complexity* [286, 287]). For the anomaly syndrome, we propose an ansatz that aims at compromising between being general enough to compress the ground states of the investigated systems while still being trainable. One way to make our circuit scalable for larger systems is to choose the number of trash qubits $n_t = \lfloor \log_2 L \rfloor$, where L is the total number of qubits. Together with the fact that our cost function is composed of only local operators, the training is expected to not suffer from Barren Plateaus [174].

Note that in principle the trash qubits can be placed anywhere in the circuit, however, when performing computations on a real quantum device it proved advantageous to explicitly take the qubit connectivity structure of the device into account in order to reduce the number of required SWAP operations. Specifically here, we placed the trash qubits in the middle of the IBM Quantum devices.

The training and inference procedure is identical to the classical anomaly detection schemes for mapping out phase diagrams [18]. In the first step, one randomly chooses a training region in the phase diagram that represents *normal* data, which is an arbitrary definition. Note that no prior knowledge about the phase diagram is therefore required. The circuit representing the anomaly syndrome is then trained on ground states of the training region, and tested on the whole phase diagram. States in the same phase as the training data are *normal* and can be disentangled, leading to a low cost. *Anomalous* states can be inferred through an increase in the cost function signaling that the corresponding ground state cannot be disentangled by the optimized circuit. From the resultant cost profile, we can deduce the phase boundary between the phase the circuit has been trained on and any other phases in the diagram. This procedure is then repeated by training in the anomalous region from the previous iteration until all phase boundaries are found.

Anomaly detection is a semi-supervised learning task. The setting is typically that one is provided with one class of data that is well known, *normal* data, and aims at finding outliers of that distribution, *anomalous* data. An archetypical example is credit card fraud where a big database of normal transactions is provided and one aims at finding fraudulent ones. We consider anomaly detection semi-supervised as labeled data (x, “normal”) is provided for training while (x, “anomalous”) is to be inferred. Here, however, we arbitrarily define (x, “normal”) and iteratively find the different classes (phases of matter). The definition of (x, “normal”) is arbitrary and does not necessitate prior knowledge. Furthermore, it is merely a means to an end to find the different classes. In that sense, the way anomaly detection is used to map out the phase diagram can be regarded as an unsupervised learning method.

Note that in previous works, where the same task has been tackled with classical machine learning techniques, it has been shown that a single ground state was sufficient to successfully train the model [62]. This feature stems from the fact that ground states within the same phase share similar properties and there is very little variance when changing the physical parameters inside one phase. We observe this feature also in the training of the VQAD.

3.3 Results

3.3.1 Simulations with ideal quantum data

In order to test the performance of VQAD, we first study the one-dimensional extended Bose Hubbard model with dimerized hoppings (DEBHM) [68],

$$\begin{aligned} H &= -\sum_{i=1}^{L-1}(J+\delta J(-1)^i)(b_i^\dagger b_{i+1}+\text{h.c.})+ \\ &\quad +\frac{U}{2}\sum_{i}^{L} n_i(n_i-1)+V\sum_{i}^{L-1} n_i n_{i+1}, \end{aligned} \tag{3.2}$$

where $b_i^\dagger(b_i)$ is the bosonic operator representing the creation(annihilation) of a particle at site i of a lattice of length L. The tunneling amplitudes $J-\delta J$ $(J+\delta J)$ indicate hopping processes on odd (even) links connecting nearest-neighbor sites, while V represents the nearest-neighbor repulsion. Here, we take the hardcore boson limit, i.e., the on-site repulsion $U/J \to \infty$, such that the local Hilbert space is two-dimensional and each site can only accommodate 0 or 1 bosons. This model can be effectively mapped into a spin-1/2 system [288].

Previous studies of the DEBHM model at half filling ($\bar{n}=0.5$) have demonstrated the existence of three distinct phases [68]. For small and intermediate values of V/J and $\delta J > 0$, we find a topological Mott insulator (TMI) displaying features analogous to a symmetry protected topological phase appearing in the dimerized spin-1/2 bond-alternating Heisenberg model [288]. On the other hand, for negative values of δJ we expect a trivial Mott insulator (MI), while in the regime where the nearest-neighbor repulsion dominates, a charge density wave (CDW) appears.

In Fig. 3.2(a)-(b), we study the phase diagram of the model in Eq. (3.2) in terms of the parameters δJ and V/J, using the density matrix renormalization group algorithm (DMRG) [36, 108, 126]. In order to differentiate between the Mott insulating phases and the CDW, one can compute the CDW order parameter

$$O_{CDW}=\sum_{i=1}^{L/2}(-1)^i\delta n_i, \tag{3.3}$$

which detects staggered patterns in the density. In Fig. 3.2(a) we report a vanishing value of O_{CDW} everywhere but in the region with large values of V/J, which corresponds to the CDW[*]. To characterize the TMI we study the entanglement spectrum (ES), which is expected to be doubly degenerate in a topologically non-trivial phase [289] due to the existence of edge states. The entanglement spectrum $\{\lambda_i\}$ is defined in terms of the positive real-valued Schmidt coefficients $\{\alpha_i\}$ of a bipartite decomposition

[*]In the definition of O_{CDW}, we consider only half of the sites of the system because the DMRG algorithm outputs a symmetric state, which is a superposition of the two degenerate ground states.

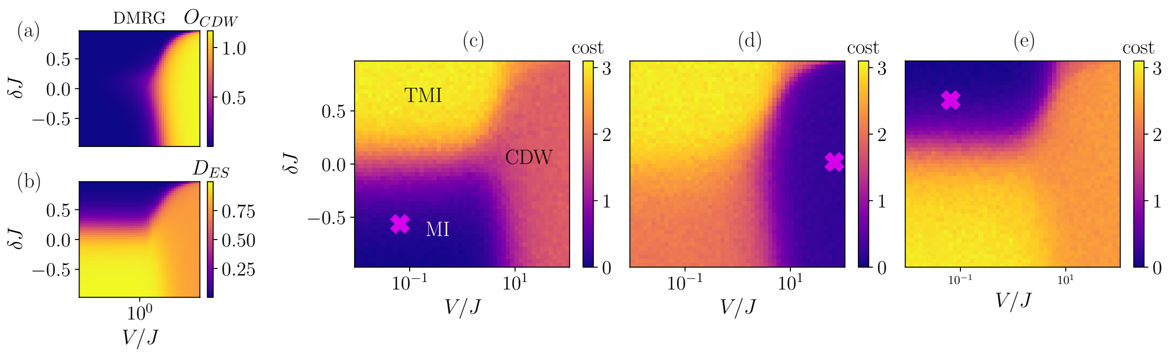


Figure 3.2: (a)-(b) Phase diagram of the DEBHM from Eq. (3.2) using (a) the order parameter O_{CDW} defined in Eq. (3.3), and (b) the degeneracy of the entanglement spectrum, D_{ES}, defined in Eq. (3.4). The results were obtained from DMRG simulations for a system of length $L = 12$ at half filling $\bar{n} = 0.5$. We fix the maximum bond dimension $BD = 50$ and the maximum number of bosons per site to $n_0 = 1$. (c)-(e) Cost/anomaly syndrome of a VQAD trained on a single ground state (indicated by a cross) of the $L = 12$ DEBHM using $n_t = 6$ trash qubits in the (c) MI phase, (d) CDW phase, and (e) TMI phase. The cost at each data point is the Hamming distance averaged over 1000 measurement shots using an ideal quantum device simulator.

of the system by $\alpha_i^2 = \exp(-\lambda_i)$. We determine its degeneracy using

$$D_{ES} = \sum_i (-1)^i e^{-\lambda_i}. \tag{3.4}$$

In Fig. 3.2(b), we show that the quantity D_{ES} vanishes only for small nearest-neighbor interaction strengths V and positive values of δJ, which correponds to the TMI. The trivial MI and CDW phases do not show a degeneracy and hence do not host topological edge states.

In the following, we test the capabilities of the VQAD with ideal states obtained from DMRG simulations. The anomaly syndrome is trained using a single representative ground state within one of the phases such that the cost measured at the trash qubits is minimised and the states of this phase can be efficiently compressed by the circuit. Afterwards, the trained circuit processes all states from the full phase diagram, ideally with similarly low cost in the same phase and significantly higher cost in other phases.

In Fig. 3.2(c)-(e) we show the resultant cost diagram for three circuits, each optimized at a different point in the phase diagram. Indeed, ground states outside of the training phase give rise to a large cost and hence are correctly classified by the VQAD as anomalous. Surprisingly, a single ground state example (indicated by the cross) was sufficient to successfully train the VQAD and infer all three phases. Similar results were recently reported for the case of classical anomaly detection using neural network auto-encoders [62].

To demonstrate the robustness of the VQAD against noise present in currently available NISQ devices we apply a depolarizing noise channel after each gate with error probabilities $p_{\mathrm{err}} = 0.001$ (single-qubit gates) and $p_{\mathrm{err}} = 0.01, 0.07$ (two-qubit gates)

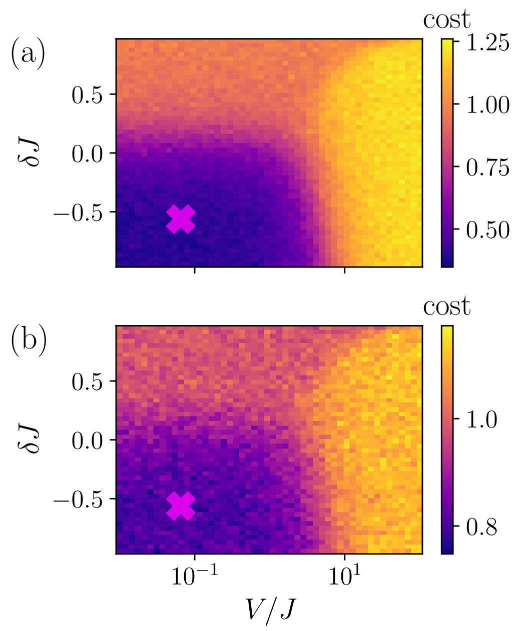


Figure 3.3: Cost of a VQAD trained on a single ground state in the MI phase (marked by the cross) of the DEBHM with $L = 12$ sites and $n_t = 2$ trash qubits. The gates of the VQAD circuit are subject to depolarizing noise with $p_{\rm err} = 0.001$ (single-qubit gates) and (a) $p_{\rm err} = 0.01$, (b) $p_{\rm err} = 0.07$ (two-qubit gates). The chosen values are motivated by the error probabilities of real devices.

and show two exemplary cost profiles of the trained anomaly detector in Fig. 3.3. Since the noise becomes more prominent with larger circuit depths, we used the two-layer VQAD circuit ansatz with only two trash qubits in this case. While it is not possible to reach a cost of zero in the training phase, the optimization still converges and all three phases can be successfully inferred. Hence, this suggests that even if the VQAD is not able to fully disentangle the trash qubits, the phase diagram can still be recovered from the resultant cost profile.

3.3.2 Experiments on a real quantum computer

We have seen that with ideal quantum data, VQAD can map out non-trivial phase diagrams including topologically non-trivial phases with and without noise in the anomaly syndrome. Next, we discuss its performance in real-noise simulations, that is with noise profiles and qubit connectivities from a real quantum device. Furthermore, we perform the quantum simulation subroutine, i.e., the ground state preparation via VQE, on the same circuit. For this task, we consider the paradigmatic transverse longitudinal field Ising (TLFI) model [66]

$$H = J\sum_{i=1}^{L} Z_i Z_{i+1} - g_x \sum_{i=1}^{L} X_i - g_z \sum_{i=1}^{L} Z_i, \tag{3.5}$$

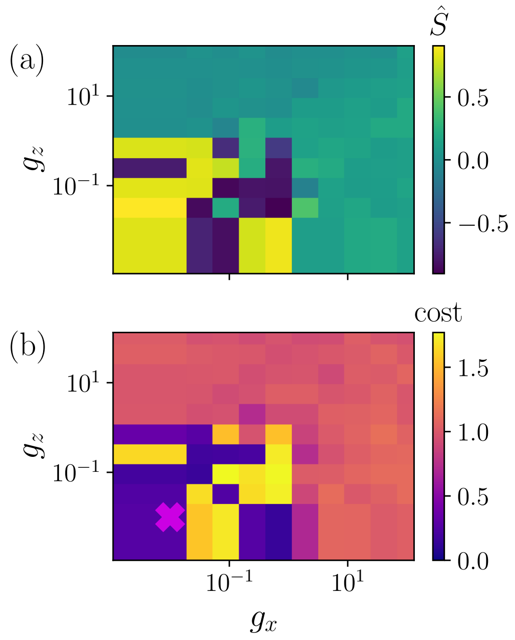


Figure 3.4: Real-noise simulations of the staggered magnetization $\hat{S}$ (a) and the anomaly syndrome (b) for the TLFI model. We trained the anomaly syndrome in the ordered phase on a state with positive $\hat{S}$, indicated by the purple cross. Inside the ordered phase, there is a perfect correlation between low cost states for positive $\hat{S}$, and very high cost where VQE converged to a negative $\hat{S}$. The paramagnetic phase is detected by a plateau in the anomaly syndrome.

where X_i, Z_i are the Pauli matrices on site i, J is the coupling strength, and g_x, g_z are the transverse and longitudinal fields, respectively. For $g_z = 0$ the model is exactly solvable and shows a quantum phase transition from a ferromagnetic (antiferromagnetic) phase for $g_x/J < 1$ and J negative (positive) to a paramagnetic one for $g_x/J > 1$ [61]. In the following we set $J = 1$ and vary the longitudinal and transverse fields. In this regime the model is not exactly solvable and the phase diagram has been extensively studied numerically [67, 290]. The antiferromagnet-paramagnet quantum phase transition is best characterized by the order parameter which in this case is the staggered magnetization

$$\hat{S} = \sum_{i=1}^{L} (-1)^i \frac{Z_i}{L}. \tag{3.6}$$

We simulate the ground states of the Hamiltonian in Eq. (3.5) using VQE for $L = 5$. On a noisy device, long-range entangling gates are performed by consecutive local two-qubit gates (SWAP operation), increasing the actual circuit depth. A large number of consecutive gates leads to decoherence due to gate errors and destroys the results. With the circuit presented in Fig. 3.1 for the VQE subroutine, we found a trade-off between expressibility and noise tolerance with a circular entanglement distribution and only one layer. Additionally, we performed measurement error mitigation [291], which can further improve the results of the cost function as seen in Fig. 3.6 in Section 3.4.

For small values of g_x and g_z, in the ferromagnetic ordered phase, the ground states

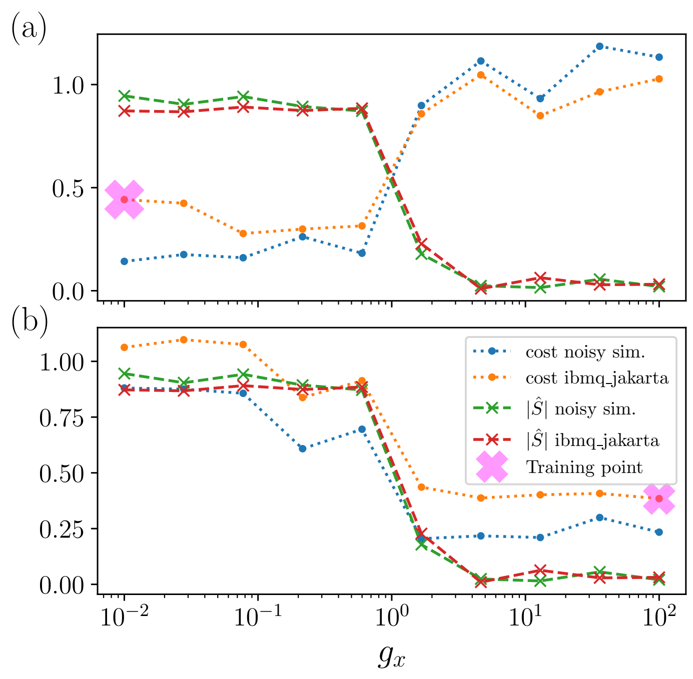


Figure 3.5: Real device VQAD experiments: We show the order parameter $\hat{S}$ compared to the VQAD results both for execution on `ibmq_jakarta` and noisy simulators with the same noise profile. We trained on a single ground state in the ordered (a) and paramagnetic (b) phase. For sampling $\hat{S}$, we use the same parameters for the VQE circuit in simulation and experiment. All values for $\hat{S}$ in the paramagnetic phase are negative, hence, for better visualization we plot its absolute value $|\hat{S}|$. For training the anomaly syndrome, the optimized parameters from the simulation are taken as an initial guess.

$\psi \simeq |10101\rangle$ ($\langle\hat{S}\rangle = 1$) and $\psi \simeq |01010\rangle$ ($\langle\hat{S}\rangle = -1$) have a similar energy, which is why the optimization can get stuck in local minima. Hence, in the ordered phase, VQE can converge to both a state with positive or negative staggered magnetization, or an equal superposition of the two as can be seen in Fig. 3.4(a). The VQAD simulation results in Fig. 3.4(b) show a perfect correlation between positive $\langle\hat{S}\rangle$ and low cost, and vice versa, negative $\langle\hat{S}\rangle$ and high cost - which, intuitively, can be expected[*]. The disordered phase is detected from the plateau of high cost (~ 1).

We see that VQAD also performs well under realistic conditions, so we next test the algorithm on a physical device. For this task, we use the $L = 5$ qubits on `ibmq_jakarta` [291]. To avoid jumps in the staggered magnetization in the ordered phase and improve convergence of the VQE optimization, we reuse already optimized parameters at neighboring points in the phase diagram as a good initial guess. Due to a large computation time overhead per execution on the real device, we additionally prepared pre-optimized parameters for both subroutines from a realistic noisy simulation, and use these as initial guesses for the optimization on the device. We found that for

[*]In a very hand-wavy way, we can understand this as we train the circuit U to perform $U|10101\rangle = |\Psi\rangle \otimes |00\rangle_{\text{trash}}$ such that $U|01010\rangle = |\Psi\rangle \otimes |11\rangle_{\text{trash}}$ if we input a state with opposite ordering.

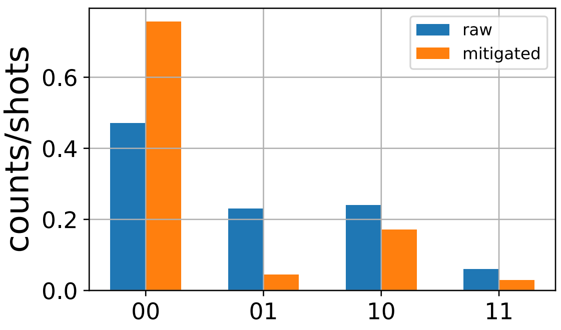


Figure 3.6: Comparison of the trash qubit measurement outcomes with and without measurement error mitigation. The anomaly syndrome circuit has been trained with and without error mitigation on a ground state of the TLFI model in the ordered phase in real-noise simulations. Ideally, all of the 1000 shots would result in the 00 bit string. By mitigating the measurement errors we improve the results towards this desired outcome.

computing the staggered magnetization it is actually not necessary to re-run the VQE optimization on the physical device, and we can achieve faithful results by directly using the optimized parameters from the simulation as seen in Fig. 3.5. The resulting cost values for the optimized circuit, plotted in Fig. 3.5, clearly distinguish the two phases, with the cost from the experiment showing solely an almost constant offset compared to the noisy simulation.

3.4 Training details

The code to run the simulations and experiments discussed above can be found in our repository on GitHub [292]. The optimization of the circuit parameters was performed using simultaneous perturbation stochastic approximation (SPSA) [167, 284]. To obtain the results presented in Fig. 3.2 of Section 3.3.1, a VQAD circuit ansatz composed of 6 layers (6 trash qubits) was employed resulting in $6L + 6$ parameters. For the noisy simulations and real-device execution discussed in Section 3.3.2, we used the ansatz in Fig. 3.1, counting $2L$ and $2L + 2$ parameters for the quantum simulation and anomaly syndrome, respectively. In classical real-noise simulations, we used 500 VQE optimization iterations for the initial ground state optimization, and 200 iterations for all subsequent optimizations where the previously optimized parameters were taken as initial guesses. For the anomaly detection circuit, we found converged results with less than 100 optimization iterations. As an example, calculating the expectation value of the magnetization takes roughly $2 - 10$ seconds on a commercial laptop (here: i7-4712HQ), while the real-device execution takes about 30 seconds. Furthermore, we used measurement error mitigation [291] provided by the Qiskit library to improve the results of the VQAD simulations in the presence of noise as illustrated in Fig. 3.6.

3.5 Conclusion

We proposed a novel quantum anomaly detection framework (VQAD) for mapping out phase diagrams of quantum many-body Hamiltonians. VQAD does not rely on labeled training data nor does it require prior knowledge about the system and its phases. We demonstrated the algorithm on the paradigmatic mixed-field Ising model and the dimerized extended Bose Hubbard model which hosts a topologically non-trivial phase. In both cases we were able to recover all distinct phases by training the VQAD ansatz on a single respective ground state. Furthermore, we tested the framework under noisy simulations and on one of the IBM Quantum devices.

Our proposed algorithm allows quantum systems simulated on a quantum computer to be analyzed end-to-end directly on the physical device without the need for measuring order parameters or for performing expensive quantum state tomography. Hence, we anticipate VQAD to become especially useful once experimental devices reach the classically intractable regime. One of the current challenges of variational quantum algorithms including VQAD are the various sources of noise and error that are present in current NISQ devices. However, with the recent progress in experimental hardware, error mitigation strategies, and circuit optimization techniques, we expect that VQAD can be applied reliably for quantum simulation tasks in the near future.

An interesting next step is to apply the VQAD framework to other systems than those discussed here (e.g., Heisenberg models, fermionic systems, time-dependent systems, etc.). Furthermore, one could investigate quantum state complexity and issues regarding the trainability of variational quantum algorithms with the VQAD model [174]. Finally, classical autoencoders allow us to interpret their latent space representation and hence gain additional insights about the system being studied [293]. Interpreting the quantum autoencoder and the information stored in the latent qubits is therefore an interesting future direction of this work.

Since the publication of this chapter, several works appeared that use supervised and unsupervised quantum machine learning techniques to study quantum systems on quantum computers in a similar fashion [294–296]. For example, Szoldra et al. [294] used our VQAD framework to identify scar states in excited quantum many-body systems. They applied it to the PXP model and a disordered, interacting spin ladder model and successfully classified these systems into families of quantum scar states that share common features. Monaco et al. [295] devised a different quantum machine learning algorithm based on quantum convolutional neural networks for mapping out phase diagrams on quantum computers. Similarly to our work, they trained their model only on a few quantum-integrable states of the full phase diagram for which an analytical solution is known. As an example they considered the axial next nearest neighbor Ising model and successfully identified all of its phases and phase boundaries. Finally, Herrmann et al. [296] also utilized a quantum convolutional neural network to detect symmetry-protected topological phases of quantum spin systems. They demonstrated that the machine learning approach leads to higher classification accuracies than a direct measurement of the string order parameter when performed on a noisy quantum device.

Apart from the quantum machine learning applications to quantum data discussed above, quantum anomaly detection and quantum autoencoders have also been applied to classical learning scenarios. Chai et al. [297] performed quantum anomaly detection on a quantum computer to identify anomalous audio samples. Park et al. [298] used quantum autoencoders for one-class classification of handwritten digits and the Fashion-MNIST data sets. They showed that their scheme outperforms other widely-used techniques like PCA and support vector machines. Lastly, Ngairangbam et al. [299] applied a quantum autoencoder-based anomaly detector to high energy physics data.

Another interesting recent result that is related to findings in our work is presented in Ref. [300]. Specifically, Caro et al. investigated the generalization capabilities of parameterized quantum circuits as a function of the training data size. Similarly to our observation that very few training examples are already sufficient for the model to learn a faithful representation, Caro et al. showed that accurate generalization is possible from a very small number of training data points. In particular, they proved that the generalization error scales as $\sqrt{T/N}$ where T is the number of parameterized gates and N is the training data size. As an example, they considered the problem of phase classification using a quantum convolutional neural network for which they numerically demonstrated that the number of required training examples is roughly independent of the number of qubits. Moreover, for the task of circuit compiling Caro et al. showed that as few as two training data points are already sufficient to obtain good generalization performances as long as the parameters are initialized close to the solution.

Chapter 4

Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks

The text in this chapter is largely based on the following publication, but extended to provide additional content.

Aikaterini Gratsea, **Friederike Metz**, and Thomas Busch
Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks
Journal of Physics A: Mathematical and Theoretical **53**, 445306 (2020)

The project emerged during an internship of Aikaterini Gratsea who I had the pleasure to supervise during her visit at OIST. I provided guidance, knowledge on reinforcement learning, and feedback. Furthermore, I contributed to the interpretation of the results, the writing of the manuscript, produced the final plots, and derived the asymptotic limit of the universally entangling coin sequence. All authors contributed to the discussions and the editing of the manuscript.

4.1 Introduction

Entanglement, the non-classical correlations between (in principle separated) particles, is harnessed in many quantum technologies nowadays and is often the key ingredient in providing an advantage over their classical counterparts [69]. As such, entanglement can be considered as a resource for many quantum information processing applications including quantum computation [99], quantum teleportation [301], quantum cryptography [302], quantum dense coding [303], and quantum metrology [304]. It is therefore not surprising that the study of entanglement generation and maximization has emerged as an important research topic.

Historically, the focus has been mostly on investigating the entanglement between multiple particles of a quantum system. However, it is also possible to consider the entanglement between two completely distinct degrees of freedom such as position and spin [305–307]. This form of entanglement has been coined hybrid entanglement and was already experimentally observed in a systems of neutrons [308] and photons [309, 310]. In principle, the entangled degrees of freedom could belong to the same particle allowing for interesting physics at the single-particle level and a resultant reduction of resources.

A platform where hybrid entanglement is naturally created are quantum walks, the quantum counterpart of the classical random walk [311–314]. Here, a particle hops along a discrete lattice in a direction dependent on its internal spin state. Quantum walks have been proposed using both a continuous-time [315] and discrete-time formulation [311]. Additionally, quantum walks can be defined on an infinite line, a cycle [316], higher dimensional lattices [317], and arbitrary graphs [318]. They have already been realized in a variety of different physical systems such as cold atoms in optical lattices [319–322], superconducting qubits [323, 324], trapped ions [325, 326], photonic platforms [327–329] and nuclear magnetic resonance systems [330, 331].

Quantum walks provide a universal model for quantum computation [70, 71] and have numerous applications within the quantum information processing domain [332]. They are used for quantum search algorithms [333, 334], quantum communication protocols [72, 73], quantum teleportation [74, 75], and quantum transport [335, 336]. Quantum walks have also recently been leveraged in both classical and quantum machine learning [337–339]. For example, quantum walk neural networks have been proposed that learn the coin operators to appropriately diffuse the information within a graph neural network [337]. Moreover, Schuld et al. [339] used stochastic quantum walks on graphs to define a quantum neural network.

The process of hybrid entanglement generation in quantum walks has been investigated in several works [340–345]. Vieira et al. [340, 341] showed that temporally disordered quantum walks can lead to maximal entanglement in the asymptotic limit of an infinitely long quantum walk. At each time step a random coin operator is sampled and used for the evolution. They also showed that the disorder-induced entanglement is independent of the initial state which is in contrast to previous works that proposed entanglement generation procedures albeit highly dependent on the initial conditions which are therefore less robust to imperfections in the state preparation [346, 347]. While the approach by Vieira et al. always leads to maximally entangled states, the large number of required time steps makes this scheme impractical for experiments with finite coherence times. Hence, it has been proposed to frame the problem of entanglement maximization as an optimization problem [344, 345]. Gratsea et al. [345] showed that using the Schmidt norm as a cost function and the coin operators at each times step as free parameters to optimize, one can achieve maximally entangled states in less than 20 steps. However, the optimal parameters were highly dependent on the initial state and required general coin operators to be implemented in an experiment.

In this work, we tackle the problem of high entanglement generation in discrete-time quantum walks using two complementary approaches. The first scheme introduces a deterministic sequence of coin operators that leads to large amounts of entanglement for a whole class of localized initial states. The coin sequences are comprised only of the Hadamard and Fourier coin, two widely used coin operators that can be readily implemented in quantum walk experiments. Furthermore, the sequences are defined for any odd number of time steps and hence, the duration of the quantum walk can be kept arbitrarily short. The initial state independence and the few number of required time steps make this scheme a robust and experimentally feasible way of creating highly entangled states in a quantum walk. Note that the dynamics of quantum walks with different deterministic coin operator sequences have already been investigated in terms of their diffusive behaviors [348–350]. Examples of previously studied coin sequences are periodic [348] and aperiodic ones such as the Fibonacci, Thue-Morse, and Rudin-Shapiro sequence [349, 350].

The second approach used within our work builds on the aforementioned idea of coin operator optimization. However, instead of optimizing the free parameters of a general SU(2) rotation matrix which might be difficult to realize experimentally, we again restrict ourselves to only the Hadamard and Fourier coins. We use reinforcement learning (RL), specifically Watkin's Q-learning algorithm [86, 351], to obtain an optimal sequence of Hadamard and Fourier coin operators that gives rise to large amounts of entanglement. Moreover, employing an RL algorithm allows us to optimize over a range of initial states such that the resultant optimal sequences produce high entanglement with only a minor dependence on the initial conditions. We find that the RL-based approach on average leads to larger Schmidt norms than the deterministic coin sequences.

In Section 4.2, I first review the notion of the one dimensional discrete time quantum walk, hybrid entanglement, and the Q-learning algorithm. In Section 4.3 I introduce the universal entangling coin sequence and present the results of the RL optimization procedure. Finally, I conclude this chapter in Section 4.4 with a brief summary of this work and subsequent developments within the field.

4.2 Background

4.2.1 Quantum walk

Discrete-time quantum walks are the quantum analog of the famous classical random walk which describes the motion of a particle on an infinite line dependent on the outcome of a coin toss. At every time step, the coin is flipped and the particle moves in one direction if the coin shows head and in the other direction if it shows tails. Similarly, the discrete-time quantum walk is also defined in terms of a particle hopping on a discrete lattice in discrete time steps (see Fig. 4.1) [311–314]. We describe the position x of the particle via a state of the infinite-dimensional Hilbert space H_w with computational basis states $\{|x\rangle : x \in \mathbb{Z}\}$. The coin degree of freedom is represented by a separate two-dimensional Hilbert space H_c spanned by $\{|\uparrow\rangle, |\downarrow\rangle\}$ which could correspond to the spin of an electron or the polarization of a photon. Hence, the full state of the particle lives in the tensor product space of the walker and coin Hilbert spaces $H = H_w \otimes H_c$.

At each step of the quantum walk, the position basis states are altered depending on the direction of the spin state, i.e., $|x, \uparrow\rangle \to |x+1, \downarrow\rangle$ and $|x, \downarrow\rangle \to |x-1, \uparrow\rangle$. The unitary operator S that performs this conditional shift takes the form

$$S = \sum_x |x-1, \uparrow\rangle \langle x, \downarrow| + |x+1, \downarrow\rangle \langle x, \uparrow| . \tag{4.1}$$

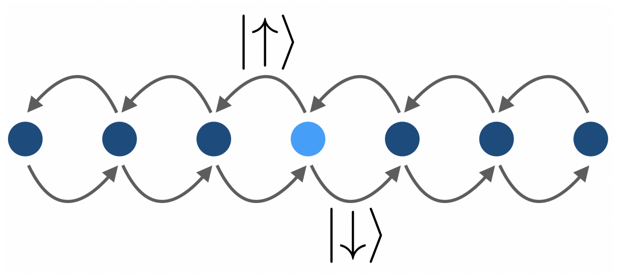


Figure 4.1: Diagrammatic representation of the 1D discrete time quantum walk. At each time step, the particle jumps to an adjacent site depending on its internal spin degree of freedom.

The coin toss is represented by a unitary operator C acting non-trivially only on the coin (spin) degrees of freedom. The role of this operator is to mix the spin states after each shift operation, or phrased differently, to produce superpositions of spin states at each distinct position. The most widely used coin operator is the Hadamard coin

$$C = I_x \otimes H_c, \qquad H_c = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{4.2}$$

which is considered unbiased since it maps each computational basis state into a uniform superposition thereof. However, in principle any SU(2) rotation can take the place of the coin operator, i.e.,

$$C = I_x \otimes e^{i\beta} \begin{bmatrix} e^{i\xi}\cos(\alpha) & e^{i\zeta}\sin(\alpha) \\ -e^{-i\zeta}\sin(\alpha) & e^{-i\xi}\cos(\alpha) \end{bmatrix}, \tag{4.3}$$

defines a valid coin operator with $\beta, \xi, \zeta \in [0, 2\pi]$ and $\alpha \in [0, \pi/2]$. Another widely used coin operator is the Fourier coin

$$F = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}. \tag{4.4}$$

A quantum walk progresses by repeatedly applying first the coin operator followed by the shift operator. If we take the initial state of the quantum walker to be $|\psi_0\rangle$ then the quantum state after t time steps will be

$$|\psi(t)\rangle = (SC)^t |\psi_0\rangle. \tag{4.5}$$

The quantum walk displays some notable differences to its classical counterpart. First of all, the evolution of the quantum state is governed by unitary dynamics which is reversible and therefore not random. Note however, that if we were to measure the quantum state after each step of the walk, the randomness introduced by the measurement process together with the collapse of the wave function would give rise to a classical random walk [311]. Hence, in what follows measurements will always be deferred to the final time step of the quantum walk in order to fully exploit quantum properties like interference and entanglement.

Another peculiar difference between the quantum and the classical random walk are the asymptotic probability distributions obtained after long times when starting with a particle centered at the origin [314]. The probability distribution of the classical random walk with an unbiased coin follows a Gaussian normal distribution with mean zero and a standard deviation scaling as $\sim \sqrt{t}$. On the other hand, the corresponding distribution of the quantum walk using the Hadamard coin is not Gaussian and gives rise to a ballistic expansion instead, i.e., the standard deviation scales with $\sim t$. Additionally, the distribution will in general be asymmetric with respect to the origin due to interference effects. The quadratically faster spreading gives the quantum walk its advantage in many of the previously discussed application areas [332].

Finally, the quantum walk also allows us to study (and harness) its quantum properties such as entanglement. In the discrete-time quantum walk entanglement naturally arises between the position and coin degrees of freedom and therefore it constitutes a unique platform to study hybrid entanglement. For example, in the case of an electron this entanglement would manifest itself between its position and spin, whereas in photonic systems one can leverage quantum walks to entangle the angular momentum with the polarization of a photon.

While any entanglement measure can be used to quantify the amount of hybrid entanglement in the system, in what follows we focus on the Schmidt norm [352]. We perform a Schmidt decomposition on the bipartite system of walker and coin

$$|\psi\rangle = \sum_{i=1}^{2} \lambda_i^{\psi} \left|\psi_w^i\right\rangle \otimes \left|\psi_c^i\right\rangle , \tag{4.6}$$

where $\lambda_i^{\psi} \geq 0$ are the Schmidt coefficients. Note that the Schmidt coefficients are equal to the eigenvalues of the reduced density matrices of either of the subsystems, e.g. $\mathrm{Tr}_w(|\psi\rangle\langle\psi|)$, and can therefore be straightforwardly computed by tracing out the position degrees of freedom. The Schmidt norm is defined as

$$\||\psi\rangle\|_p := \left(\sum_{i=1}^{2} \left(\lambda_i^{\psi}\right)^p\right)^{1/p}, \qquad (p \geq 1). \tag{4.7}$$

In the subsequent work we set $p = 1$. A maximally entangled state corresponds to a Schmidt norm of $\sqrt{2}$, while a non-entangled state gives rise to a value of 1.

4.2.2 Q-learning

We implement the off-policy Q-learning algorithm with the goal of maximizing the hybrid entanglement between the walker and the coin degree of freedom [351]. Each training episode consists of a fixed number of time steps n of the quantum walk. Since we are interested in maximizing the entanglement after the evolution is complete, we set all rewards at intermediate time steps to zero and allow for a nonzero reward only at the final time step, which we set to the Schmidt norm.

At each time step of the quantum walk, the agent can choose between two actions defined as $A \in \{H, F\}$, where H and F correspond to the Hadamard and Fourier coin operator respectively (see Eq. (4.4)). This choice is made after obtaining information about the current state of the environment. One way of defining the RL state would be to use the full quantum state of the system at each time step of the quantum walk. However, since the quantum state is essentially a vector of continuous complex numbers, it cannot be straightforwardly employed in tabular (discrete) RL settings and more sophisticated methods like neural network function approximators are needed [86]. However, in our case we can use the fact that the dynamics of the system are deterministic and therefore the history of actions (applied coins) contains the same information for a fixed initial state. Specifically, for a given number of time steps n and a specific initial state ψ_0, there are 2^n possible sequences. For example, for the case $n = 2$ the complete set of sequences are $\{HH\psi_0, HF\psi_0, FH\psi_0, FF\psi_0\}$. Hence, no information about the intermediate physical states is needed and the RL states are simply given by $S \in \{\text{init}, H, F, HH, HF, FH, FF\}$. Here, *init* refers to the initial state of the environment before the quantum walk evolution has started.

For obtaining the optimal policy $\pi^*(S) = A$, which indicates the optimal action to take given the current state, we employ the Q-Learning algorithm which is based on learning an optimal Q function. The Q value $Q^\pi(S, A)$ of a state-action pair is defined as the expected cumulative future reward when starting in state S, taking action A, and following the policy π thereafter

$$Q^\pi(S, A) \doteq \mathbb{E}_\pi \left[\sum_i R_i \middle| S, A \right]. \tag{4.8}$$

Therefore, the Q value $Q(S, A)$ is a measure of how promising it is to choose the respective action A in a state S. The optimal Q value $Q^*(S, A)$ is simply defined as the maximum Q value over all policies $Q^*(S, A) = \max_\pi Q^\pi(S, A)$. In case the optimal Q function is known for all state-action pairs, the optimal policy can be inferred by selecting actions that maximize the Q value, i.e., a greedy action selection

$$\pi^*(S) = \arg\max_A Q^*(S, A). \tag{4.9}$$

Hence, it suffices to learn the optimal Q values which can be achieved through an iterative update rule known as Temporal Difference learning

$$Q(S_i, A_i) \to Q(S_i, A_i) + \alpha \left[R_i + \max_A Q(S_{i+1}, A) - Q(S_i, A_i) \right], \tag{4.10}$$

where $\alpha \in [0, 1]$ is the learning rate and the term in the brackets is called the target. It can be shown that the Q values eventually converge to their optimal values if the policy that is followed during training has a finite probability of visiting all state-action pairs [86]. Here, we use an ϵ-greedy action selection during training, i.e., the agent acts randomly with probability ϵ and otherwise takes action A_i which maximizes the Q value in the current state: $A_i = \text{argmax}_A Q(S_i, A)$. Moreover, for a better trade-off between exploration of the full action space and exploitation of high rewards, ϵ is exponentially

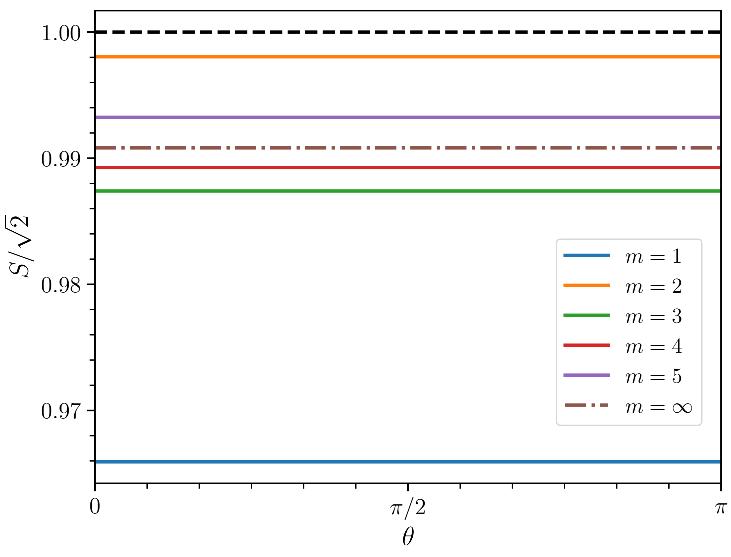


Figure 4.2: Schmidt norm S computed after evolution with the sequence seq$^*(2m+1) = [(H,F)^m, F]$ for $m = 1, ..., 5$ as a function of the initial state parameter θ when $\phi = 0$. The black dashed line indicates the maximum achievable value and the brown dashed-dotted line the asymptotic value for $m \to \infty$.

decaying after each training episode i

$$\epsilon(i) = (\epsilon_{\text{init}} - \epsilon_{\text{fin}}) \exp\left[\frac{-8i}{N_{\text{episodes}}}\right] + \epsilon_{\text{fin}}, \tag{4.11}$$

with ϵ_{init} and ϵ_{fin} being the initial and final value of ϵ, respectively. The exponential decay ensures that at the beginning of training the agent acts mostly random and explores a variety of different actions while towards the end of training actions are chosen more deterministically according to the target policy. Once training has successfully converged, the optimal policy is given by a fully greedy action selection given through Eq. (4.9).

4.3 Hybrid entanglement creation

4.3.1 Universal entangling coin sequence

We are interested in generating highly entangled states during a quantum walk independent of the initial state. Since the final amount of entanglement cannot be fully independent for all possible initial states [346, 347], we restrict the initial state to the class of localized states

$$|\psi_0\rangle = \cos(\theta/2)\,|0,\uparrow\rangle + e^{i\phi}\sin(\theta/2)\,|0,\downarrow\rangle\,, \tag{4.12}$$

with zero relative phase ($\phi = 0$). Hence, the problem reduces to finding a sequence of coin operators in time that generates entanglement independent of the initial state

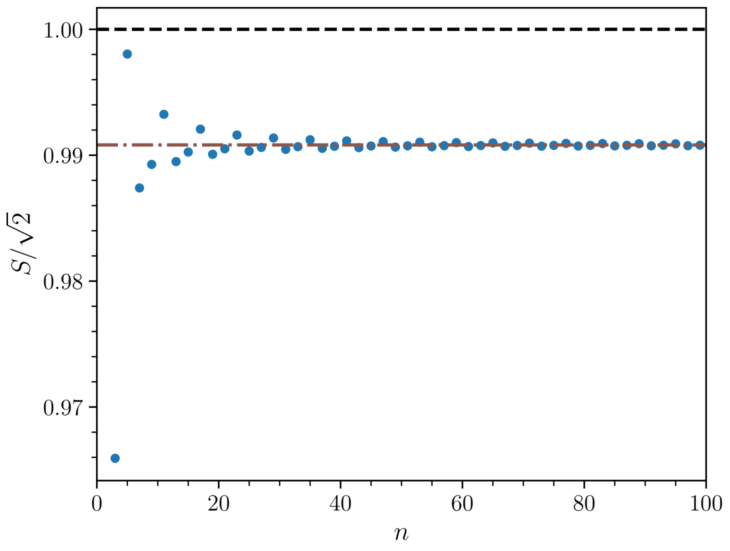


Figure 4.3: Schmidt norm S after evolution with the sequence $\mathrm{seq}^*(2m+1) = [(H,F)^m, F]$ as a function of the number of steps $n = 2m+1$ (only odd time steps are displayed). Each point is an average over 1000 random initial states with $\phi = 0$. The variances calculate to zero suggesting that the sequences seq* generate states with an amount of entanglement being independent of θ. The dashed line denotes again the maximum achievable Schmidt norm while the brown dashed-dotted line indicates the asymptotic value reached for $n = (2m+1) \to \infty$.

parameter θ. For this we propose a sequence given by $\mathrm{seq}^*(2m+1) = [(H,F)^m, F]$, $m \in \mathbb{Z}$ for a quantum walk with $2m+1$ time steps. This sequence consists of an alternating application of the Hadamard and Fourier coin with an additional Fourier coin applied at the final time step and hence always describes a quantum walk with odd number of steps. In Fig. 4.2 we plot the Schmidt norm at the end of the quantum walk evolution with the proposed sequence for several different time steps as a function of the parameter θ. One can easily see that the value of entanglement is always very close to the maximal amount possible and indeed independent of θ for each sequence. However it depends on the number of steps taken for short sequences, but quickly converges to a value close to $S/\sqrt{2} = 0.99$ for larger values of n (see Fig. 4.3). The derivation of the asymptotic limit of this sequence is shown in Section 4.3.2. Each point in Fig. 4.3 is obtained after averaging over 1000 random angles θ and the zero variances confirm that the Schmidt norm is independent of the parameter θ. Therefore, from now on we will refer to the sequence seq* as a universal entangler for the class of initial states defined by $\phi = 0$.

In the following we will give an intuitive explanation of how the universal behavior emerges from this sequence. Generally, the Schmidt norm can be calculated from the reduced density matrix of the coin degree of freedom after tracing out the walker states. Representing the reduced density matrix ρ as a vector on the Bloch sphere

$$\rho = \frac{1}{2}I + \vec{\alpha}\vec{\sigma}, \tag{4.13}$$

where $\vec{\alpha}$ is the Bloch vector and $\vec{\sigma}$ is a vector of Pauli matrices, the Schmidt norm can be expressed in the form

$$S = \sqrt{\frac{1}{2} + \mid \vec{\alpha} \mid} + \sqrt{\frac{1}{2} - \mid \vec{\alpha} \mid}, \tag{4.14}$$

which only depends on the norm of the Bloch vector $\vec{\alpha}$. During the evolution with the universal entangling sequence, the behavior of the Bloch vector $\vec{\alpha}$ follows a periodic pattern. Specifically, after each application of the Hadamard operator the Bloch vector points along the x-axis, while the subsequent application of the Fourier operator projects it onto the y-axis. For example, the sequence $[H, F, H]$ gives rise to $\vec{\alpha_3} = ((\cos\theta + \sin\theta)/4, 0, 0)$, whereas after the sequence $[H, F, H, F]$ we obtain $\vec{\alpha_4} = (0, (-\cos\theta + 4\sin\theta)/16, 0)$. At the end of the time evolution, an additional Fourier coin is applied, which rotates the Bloch vector into a θ-dependent direction in the x-y plane with a norm that is independent of θ. For example, after the evolution with the sequence $\text{seq}^*(5) = [H, F, H, F, F]$, the Bloch vector calculates to $\vec{\alpha_5} = (\cos\theta/16, \sin\theta/16, 0)$ with $\mid \vec{\alpha_5} \mid = 1/16$ and the Schmidt norm is independent of θ and approximately equal to 1.4114. The same property is also observed in the asymptotic limit when $m \to \infty$ (see Section 4.3.2).

In order to better understand the behavior of the universal entangling sequence, we explore the role of the two coin operators H and F. The Fourier operator seems to be of significant importance for generating highly entangled states. Generally, it increases the localization of the quantum state [353] which has been associated with an enhancement in the entanglement [343]. On the other hand, the Hadamard operator belongs to the class of rotation matrices [354] and we have found that replacing it with a more general unbalanced operator does not change the universal behavior of the sequence. The generalized Hadamard operator $\tilde{H}$ is given by

$$\tilde{H}(\omega) = \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ \sin(\omega) & -\cos(\omega) \end{bmatrix}, \tag{4.15}$$

so that the sequence takes the new form of $[(\tilde{H}(\omega), F)^m, F]$. Figure 4.4 shows the Schmidt norm after a 5, 7, and 15 step quantum walk as a function of the parameter ω for initial states with zero relative phase. Each data point was obtained after averaging over 1000 random angles θ of the initial state and the variance again calculates to zero in all cases. Therefore the amount of entanglement created is still independent of θ. Moreover, the plot suggests that by properly choosing the parameter ω for a given length of the sequence, the performance of the universal entangling sequence can be improved and a state close to a maximally entangled state can be reached.

Let us finally note that the effect of a nonzero relative phase ϕ in the initial state can be cancelled out in two ways using the phase operator Z given by

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\phi} \end{bmatrix}. \tag{4.16}$$

The phase operator can be applied either directly to the initial state or to the coin operators. In the latter case, the H and F operators are altered to HZ and FZ, re-

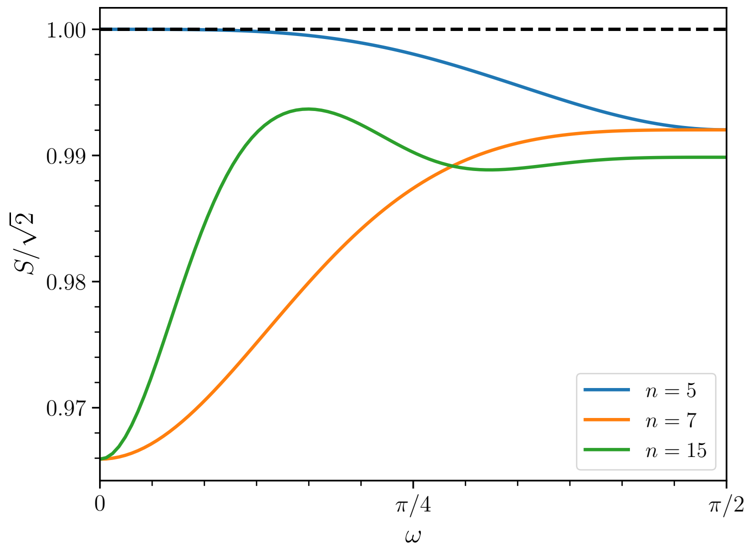


Figure 4.4: Value of the Schmidt norm as a function of the generalized Hadamard operator parameter ω after the sequence $[(\tilde{H}(\omega), F)^m, F]$ for a 5-step (blue), 7-step (orange) and 15-step (green) quantum walk. Each point is an average over 1000 random initial states with $\phi = 0$. The variances calculate to zero and the dashed line indicates the maximum achievable Schmidt norm.

spectively. However, this requires that the relative phase of the initial state is known beforehand, which can be the case if the creation process of the initial state is deterministic.

4.3.2 Asymptotic limit of the universal entangling coin sequence

In the following we derive the asymptotic limit of the coin reduced density matrix under the universal entangling sequence, i.e. $\text{seq}^*(2m+1) = [(H, F)^m, F]$ with $m \to \infty$, which allows us to calculate the asymptotic value of the Schmidt norm and prove its independence of the initial state angle θ. We follow the approach of Refs. [355, 356] where the evolution of the reduced density matrix of the coin degree of freedom is directly computed through an effective superoperator in Fourier space.

The quantum walk shift operator S of Eq. (4.1) can be expressed in momentum space after performing a Fourier transform defined by $|k\rangle = \sum_x e^{ikx}|x\rangle$, which leads to

$$S_k = |k\rangle\langle k| \otimes \left(e^{-ik}|\downarrow\rangle\langle\uparrow| + e^{ik}|\uparrow\rangle\langle\downarrow|\right). \tag{4.17}$$

The combined effect of the shift and coin operator can therefore be reduced to a 2×2 matrix acting on the coin degree of freedom only and for the Hadamard and Fourier coin we obtain

$$S_k H = \mathbb{I}_k \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} e^{ik} & -e^{ik} \\ e^{-ik} & e^{-ik} \end{pmatrix}, \tag{4.18}$$

$$S_k F = \mathbb{I}_k \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} ie^{ik} & e^{ik} \\ e^{-ik} & ie^{-ik} \end{pmatrix}. \tag{4.19}$$

These operators act on the full quantum state $|\psi\rangle$ (coin and momentum degree of freedom), however, we can also directly work in the reduced space of the coin which can be represented as a vector on the Bloch sphere as

$$\rho = \mathrm{Tr}_k \left(|\psi\rangle \langle\psi| \right) = \alpha_0 I + \alpha_1 \sigma_1 + \alpha_2 \sigma_2 + \alpha_3 \sigma_3. \tag{4.20}$$

For an arbitrary initial state of Eq. (4.12) the Bloch vector components yield

$$\vec{\rho}_0 = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ \cos\varphi \sin\theta \\ -\sin\varphi \sin\theta \\ \cos\theta \end{pmatrix}. \tag{4.21}$$

During the quantum walk evolution the reduced density matrix transforms according to an effective evolution superoperator L_k and after n steps of the quantum walk is given by

$$\rho_n = \int_{-\pi}^{\pi} \frac{dk}{2\pi} \left(L_k\right)^n \rho_0. \tag{4.22}$$

Using the vector notation of Eq. (4.21), the operator L_k can be represented as a 4×4 matrix. The matrix entries are obtained after working out how each of the Pauli matrices transforms under the combined effect of shift and coin operator, i.e., Eqs. (4.18) and (4.19). For the case of the Hadamard and Fourier coin the superoperators compute to

$$L_k^H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \sin 2k & \cos 2k \\ 0 & 0 & \cos 2k & -\sin 2k \\ 0 & -1 & 0 & 0 \end{pmatrix}, \tag{4.23}$$

$$L_k^F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2k & 0 & -\sin 2k \\ 0 & -\sin 2k & 0 & -\cos 2k \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{4.24}$$

Hence, two steps of the quantum walk with a Hadamard coin applied at the first time step and a Fourier coin applied at the second time step, give rise to the following superoperator

$$L_k^{HF} = L_k^F L_k^H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sin 2k & \sin 2k \cos 2k & \cos^2 2k \\ 0 & \cos 2k & -\sin^2 2k & -\sin 2k \cos 2k \\ 0 & 0 & \cos 2k & -\sin 2k \end{pmatrix}. \tag{4.25}$$

Since we are interested in the long time behavior, we first diagonalize the matrix above before exponentiating it to the desired power. The eigenvalues are given by

$$\lambda_0 = 1,\ \lambda_1 = 1,\ \lambda_2 = e^{i(\gamma+\pi)},\ \lambda_3 = e^{-i(\gamma+\pi)}, \tag{4.26}$$

with

$$\cos\gamma = \frac{1}{2}(1+\sin^2 2k). \tag{4.27}$$

After $n = 2m$ steps of the quantum walk with a Hadamard and Fourier coin applied at alternating time steps we obtain

$$\left(L_k^{HF}\right)^m = B \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{im(\gamma+\pi)} & 0 \\ 0 & 0 & 0 & e^{-im(\gamma+\pi)} \end{pmatrix} B^\dagger, \tag{4.28}$$

where the matrix B contains the corresponding eigenvectors as column entries

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & v_{11} & v_{12} & v_{13} \\ 0 & v_{21} & v_{22} & v_{23} \\ 0 & v_{31} & v_{32} & v_{33} \end{pmatrix}. \tag{4.29}$$

When taking the limit $m \to \infty$, the oscillatory terms $e^{\pm im(\gamma+\pi)}$ vanish due to the stationary phase theorem. Therefore, we get the following expression for the asymptotic superoperator

$$\left(L_k^{HF}\right)^m \xrightarrow[m\to\infty]{} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & |v_{11}|^2 & v_{11}v_{21}^* & v_{11}v_{31}^* \\ 0 & v_{21}v_{11}^* & |v_{21}|^2 & v_{21}v_{31}^* \\ 0 & v_{31}v_{11}^* & v_{31}v_{21}^* & |v_{31}|^2 \end{pmatrix}, \tag{4.30}$$

which only involves the components of the first eigenvector

$$\vec{v_1} = \begin{pmatrix} v_{11} \\ v_{21} \\ v_{31} \end{pmatrix} = \frac{\cos 2k}{\sqrt{4-(\sin^2 2k+1)^2}} \begin{pmatrix} 1+\sin 2k \\ \cos 2k \\ 1-\sin 2k \end{pmatrix}. \tag{4.31}$$

The superoperator L_k^* of the universal entangling sequence is obtained by acting with an additional final Fourier superoperator L_k^F

$$L_k^*(m) = L_k^F \left(L_k^{HF}\right)^m . \tag{4.32}$$

The asymptotic limit of the reduced density matrix can then be calculated by performing the momentum integrals for each matrix entry separately giving rise to

$$\begin{aligned}
\vec{\rho}_\infty^* &= \lim_{m\to\infty} \int_{-\pi}^{\pi} \frac{dk}{2\pi} L_k^*(m)\vec{\rho}_0 \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1+\frac{2}{\sqrt{3}} & 2-\sqrt{3} \\ 0 & -2+\sqrt{3} & 1-\frac{2}{\sqrt{3}} & 0 \\ 0 & 0 & -1+\frac{2}{\sqrt{3}} & 0 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} 1 \\ \left(2-\sqrt{3}\right)\cos\theta \\ \left(-2+\sqrt{3}\right)\sin\theta \\ 0 \end{pmatrix} .
\end{aligned} \tag{4.33}$$

In the last line we used that $\phi = 0$ for the initial states considered here. The final state lies in the $x - y$ plane of the Bloch sphere with a norm independent of the angle θ. As a consequence the Schmidt norm defined in Eq. (4.14), which is only a function of the length of the Bloch vector, is also independent of θ and computes to

$$\begin{aligned}
S &= \sqrt{\frac{1}{2} + \frac{1}{2}(2-\sqrt{3})} + \sqrt{\frac{1}{2} - \frac{1}{2}(2-\sqrt{3})} \\
&\sim 0.9908 \times \sqrt{2}.
\end{aligned} \tag{4.34}$$

This value matches the asymptotic behavior we observe in Fig. 4.3.

4.3.3 Optimal coin sequences

Let us next address the question whether we can find coin sequences that perform better on average than the universal entangling sequence, i.e., that generate higher values of entanglement across all initial states. To solve this optimization problem efficiently we employ the Q-Learning algorithm described in Section 4.2.2. We should emphasize that for a given number of steps n, the goal is to find the optimal sequence of coins out of the 2^n possible sequences that maximizes the Schmidt norm (the reward) for all initial states. Our RL framework allows us to solve for this objective due to the agents ignorance of the quantum state. Even though different initial quantum states are used for each episode, the agent has access only to the states defined by the history of actions and hence no information about the quantum state is used for training.

For a better comparison to the previous section, we again restrict the initial states to a subspace defined by $\phi = 0$. For each episode of training, the remaining initial

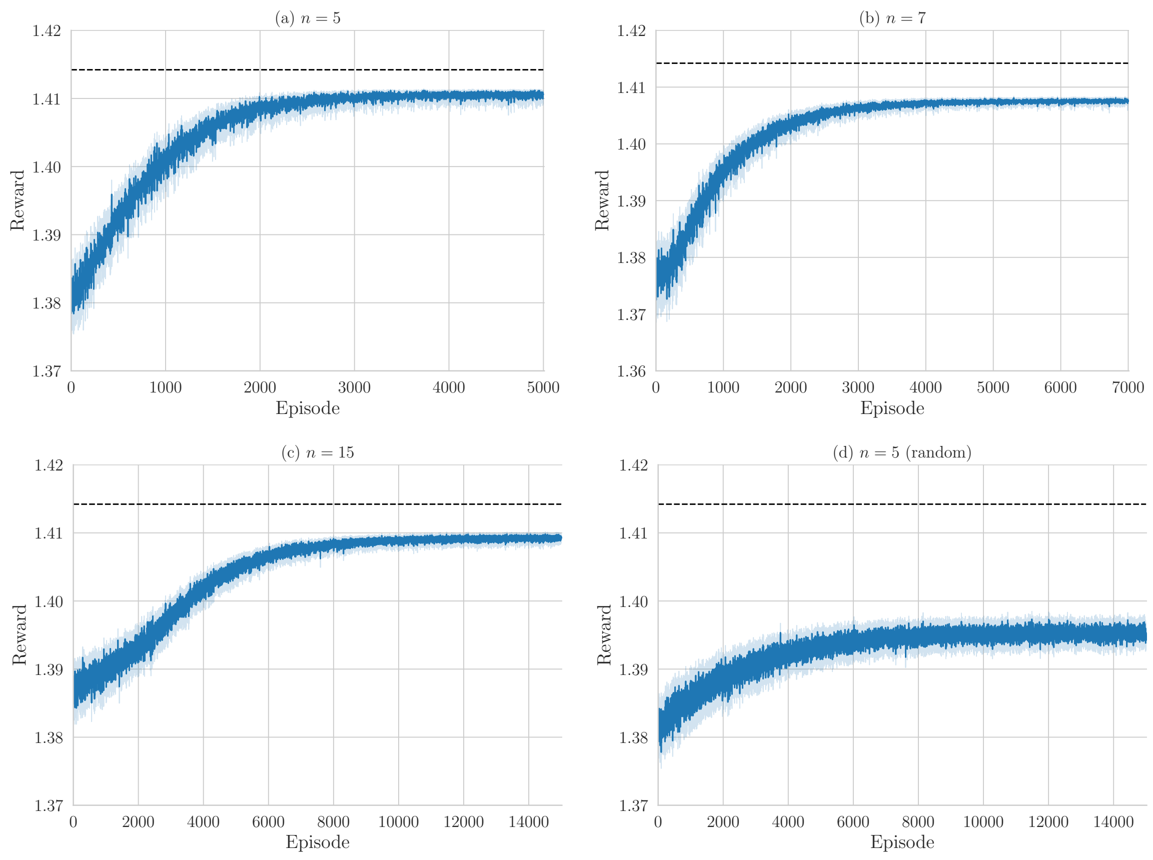


Figure 4.5: Learning curves for the optimization problems. The episodic reward (Schmidt norm) is averaged over 300 ((a), (b)) or 400 ((c), (d)) independent runs. The light blue area corresponds to the confidence interval and dashed lines denote the maximally achievable reward of $\sqrt{2}$. (a)-(c) Learning curves for the 5, 7, and 15 step quantum walk where the initial state parameter ϕ is set to zero and the parameter θ is sampled from a uniform distribution at the beginning of each new episode. (d) Learning curve for the 5 step quantum walk where both initial state parameters ϕ and θ are sampled at the beginning of each training episode.

state parameter θ is sampled from a uniform distribution such that each episode is initialized with a different quantum state. All instances of training were performed using the Q-Learning algorithm [351] with Q values initialized to zero. We found a learning rate of $\alpha = 0.7$ to give the best results overall. The exploration parameter ϵ decays exponentially throughout the training from an initial value of $\epsilon_i = 0.9$ to a final value of $\epsilon_f = 0.01$.

As an example we show the learning curves and results of the RL optimization obtained for a 5, 7, and 15 step quantum walk in Figs. 4.5-4.6. The Schmidt norm achieved by the optimal sequence is plotted as a function of the parameter θ. Dashed lines of the same color correspond to the respective universal entangling sequence from the last section. Notice that in the case of a 5 step quantum walk the universal sequence and the optimal sequence coincide, i.e., the RL agent finds $[H, F, H, F, F]$ to be optimal. For the cases of a 7 and 15 step quantum walk the optimal sequences differ from the

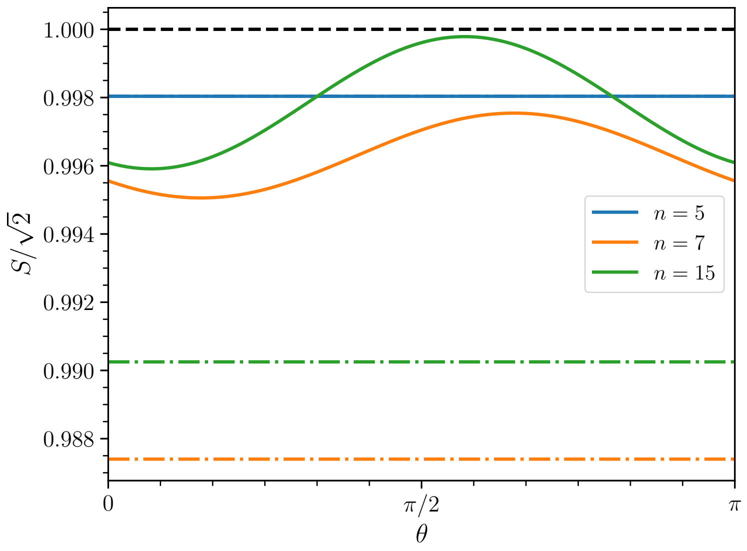


Figure 4.6: Schmidt norm reached after an evolution of 5-step (blue), 7-step (orange), and 15-step (green) quantum walk with the optimal sequence (solid line) and the universal entangling sequence (dotted dashed line). The black dashed line denotes the maximum achievable Schmidt norm. In the case of a 5-step quantum walk the optimal and universal entangling sequence coincide. The optimal sequences are $[H, F, H, F, F]$, $[F, H, H, H, F, H, H]$, and $[F, H^7, F, H^6]$ respectively and were obtained using the Q-Learning algorithm.

universal ones and the obtained Schmidt norm is not independent of the initial state anymore. However, in both cases the amount of entanglement exceeds that of the universal sequence for all initial state parameters θ.

In order to validate the result, we compared the reinforcement learning algorithm with a simple brute-force method for the case of the 5 step quantum walk. The brute-force algorithm explores all of the possible $2^5 = 32$ coin sequences for 1000 random initial states and computes the average Schmidt norm for each sequence. We find that the policy giving rise to the highest average entanglement is indeed the sequence the RL algorithm suggested previously: $[H, F, H, F, F]$. While for quantum walks with only a few steps a simple brute-force method as described above is able to identify optimal policies, the RL algorithm becomes advantageous for larger numbers of time steps. The number of possible coin sequences grows exponentially with the number of steps and hence quickly becomes intractable by any brute-force method.

Finally, we train an RL agent on completely random initial states, where both ϕ and θ are uniformly sampled at the beginning of each episode. For a five step walk the optimal sequence suggested by the RL agent is $[F, F, H, H, H]$ and in Fig. 4.7 we show the values of the achieved Schmidt norm as a function of the initial state parameters. One can see that the final amount of entanglement depends slightly stronger on the initial state compared to the previous cases where we only considered initial states with $\phi = 0$. This is not surprising since it is known that quantum walks of only a few steps cannot generate highly entangled states in a fully universal way for all initial states at the same time [346, 347]. However, the RL algorithm is still able to identify a sequence

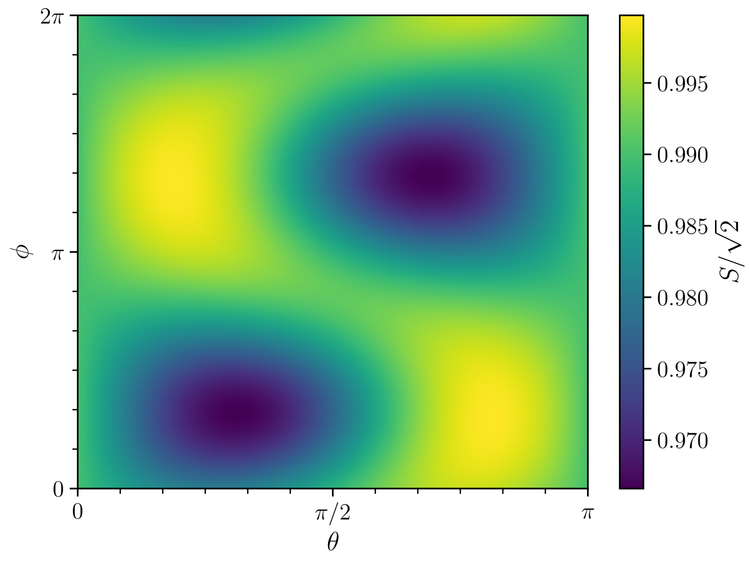


Figure 4.7: Schmidt norm obtained by evolving with the optimal policy $[F, F, H, H, H]$ as a function of the initial state parameters ϕ and θ.

that, at least on average, performs better than others.

4.4 Conclusion

We proposed two different schemes for creating highly entangled states in a 1D discrete time quantum walk. First, we showed that a specific sequence of Hadamard and Fourier coin operators leads to large Schmidt norms independent of the localized initial state. The sequences are composed of an alternate application of Hadamard and Fourier coins with an additional Fourier coin applied at the final time step. The sequences are defined for any odd number of steps and hence, the overall quantum walk time duration can be kept short which is important for many experimental realizations. Moreover, we demonstrated that by replacing the Hadamard coin with arbitrary (fixed) rotation operators allows the final entanglement to be tuned further while maintaining the universal-entangling property. Finally, we derived an expression for the asymptotic state of the deterministic coin sequences, i.e., in the limit of an infinitely-long quantum walk, and computed the corresponding Schmidt norm that would be asymptotically attained.

In the second approach, we used a reinforcement learning (RL) algorithm to maximize the hybrid entanglement over different sequences of Hadamard and Fourier coin operators. Employing an RL-based approach over other search algorithms is advantageous since the space of possible sequences grows exponentially in the number of time steps and thus, a brute-force optimization would quickly become intractable. Moreover, the proposed algorithm allows us to find optimal coin sequences for a whole class of different initial states. Therefore, our entanglement generation sequences are more robust to imperfections in the initial state preparation process. The obtained optimal protocols for different number of steps on average give rise to larger Schmidt norms than the deterministic sequences. However, the resultant values vary with the initial state parameters in contrast to the previous case.

Our work has motivated further research in the field of entanglement generation in quantum walks and has already been cited several times since its publication. Tao et al. [357] experimentally studied the hybrid entanglement in a photonic quantum walk system. They compared different approaches of creating highly entangled states and verified that by using optimized coin operators maximal entanglement can be reached in a few steps only. Zhang et al. [358] proposed a position-inhomogeneous quantum walk in which a position-dependent coin operator involving a parameterized phase is used at every time step. They showed that maximal entanglement can be achieved for any odd number of quantum walk steps. Furthermore, Zhang et al. demonstrated an experimental realization of their proposal within an optical network setup. Finally, Naves et al. [359] considered a quantum walk with a time-dependent shift operator referred to as elephant quantum walk, where the position step size is randomly sampled at every time step. It was shown that for certain step size distributions, the generalized elephant quantum walk gives rise to maximally entangled states independent of the localized initial state and coin operator.

Our work has been one of the first that applies classical machine learning tools to quantum walks. Shortly after our publication, another paper appeared in which supervised learning is used to estimate coin operator parameters in a quantum walk from its final distribution [360]. Additionally, our approach of utilizing reinforcement learning to obtain universal optimal protocols for a range of initial conditions, has been picked up in other areas outside the domain of quantum walks. For example, He et al. [31] applied reinforcement learning to the problem of state preparation in semiconductor double quantum dots and demonstrated its capabilities of devising protocols for arbitrary initial states.

Chapter 5

Self-correcting quantum many-body control using reinforcement learning with tensor networks

The text in this chapter is largely based on the following preprint article, but extended to provide additional content.

Friederike Metz and Marin Bukov
Self-Correcting Quantum Many-Body Control using Reinforcement Learning with Tensor Networks
arXiv:2201.11790 [quant-ph] (2022)

I performed the numerical simulations, the theoretical analysis, and wrote a first version of the manuscript. Marin Bukov supervised the work. All authors contributed to the discussions, the interpretation of the results, and the editing of the manuscript draft.

5.1 Introduction

Quantum many-body control is an essential prerequisite for the reliable operation of modern quantum technologies which are based on harnessing quantum correlations. For example, quantum computing often involves high-fidelity state manipulation as a necessary component of most quantum algorithms [167, 279]; In quantum simulation, the underlying AMO platforms require to prepare the system in a desired state before its properties can be measured and studied [77–79]; And quantum metrology relies on the controlled engineering of (critical) states to maximize the sensitivity to physical parameters [361, 362]. Many-body control can also be considered in its own right, as a numerical tool which offers insights into concepts such as quantum phases and phase transitions [363]. Moreover, it can reveal novel theoretical phenomena such as phase transitions in the control landscape [364], and bears a direct relation to our understanding of quantum complexity [365].

Compared to single- and few-particle physics, working in the quantum *many-body* domain introduces the formidable difficulty of dealing with an exponentially large

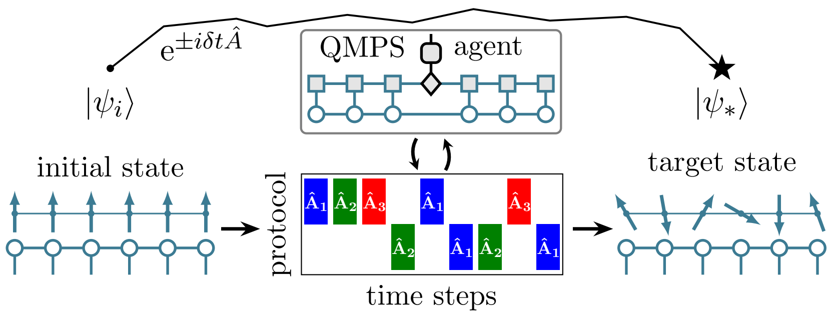

$e^{\pm i\delta t\hat{A}}$
$|\psi_i\rangle$
QMPS agent
$|\psi_*\rangle$
initial state
protocol
$\hat{A}_1$ $\hat{A}_2$ $\hat{A}_3$ $\hat{A}_2$ $\hat{A}_1$ $\hat{A}_1$ $\hat{A}_2$ $\hat{A}_3$ $\hat{A}_1$
target state
time steps

Figure 5.1: Illustration of the QMPS framework for state preparation. The optimized QMPS agent outputs a control protocol as a sequence of operators $\hat{A}_j$, which time evolve the initial spin state into the desired target state (marked by a star).

Hilbert space. A specific manifestation is the accurate description and manipulation of quantum entanglement shared between many degrees of freedom. This poses a limitation for classical simulation methods, since memory and compute time resources scale exponentially with the system size. Fortunately, there exists a powerful framework to simulate the physics of one-dimensional (1d) quantum many-body systems, based on matrix product states (MPS) [36, 37, 113, 126]. MPS provide a compressed representation of many-body wave functions and allow for efficient computation with resources scaling only linearly in the system size for area-law entangled states [123, 124].

While MPS-based algorithms have been used in the context of optimal many-body control to find high-fidelity protocols that manipulate interacting ultracold quantum gases [366–368], the advantages of deep reinforcement learning (RL) for quantum control, have so far been investigated using exact simulations of only a small number of interacting quantum degrees of freedom. Nevertheless, policy-gradient and value-function RL algorithms have recently been established as useful tools in the study of quantum state preparation [16, 29–32, 205–213], quantum error correction and mitigation [17, 22–24], quantum circuit design [25–28], and quantum metrology [214, 215]; quantum reinforcement learning algorithms have been proposed as well [136, 369–372]. Thus, in times of rapidly developing quantum simulators which exceed the computational capabilities of classical computers [373], the natural question arises regarding scaling up the size of quantum systems in RL control studies beyond exact diagonalization methods.

In this work, we develop a new deep RL framework for quantum many-body control, based on MPS in two complementary ways. First, we adopt the MPS description of quantum states: this allows us to control large interacting 1d systems, whose quantum dynamics we simulate within the RL environment. Second, representing the RL state in the form of an MPS, naturally suggests the use of tensors network as (part of) the deep learning architecture for the RL agent, e.g., instead of a conventional neural network (NN) ansatz. Therefore, inspired by earlier examples of tensor-network-based machine learning [33, 42, 144], we approximate the RL agent as a hybrid MPS-NN network, called QMPS. With these innovations at hand, the required computational resources scale linearly with the system size, in contrast to learning from the full many-body wave function. Ultimately, this allows us to train an RL agent to control a larger number

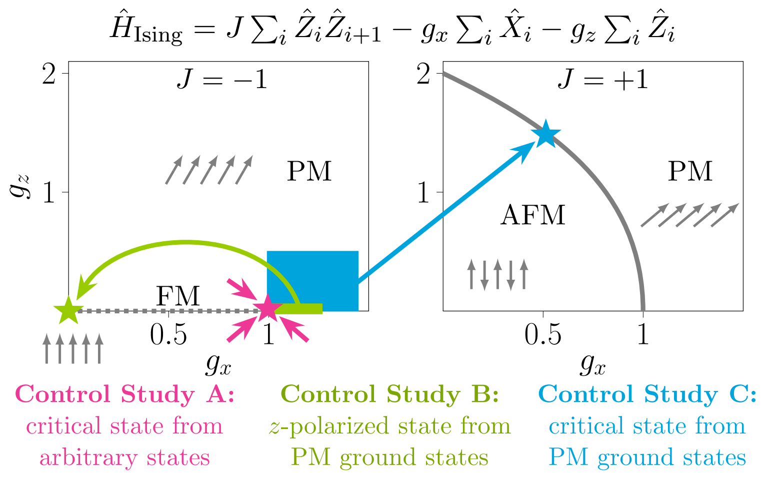

$\hat{H}_{\text{Ising}} = J\sum_i \hat{Z}_i\hat{Z}_{i+1} - g_x\sum_i \hat{X}_i - g_z\sum_i \hat{Z}_i$
$J=-1$
$J=+1$
PM
FM
PM
AFM
g_z
g_x
g_x
2
1
0.5
1
2
1
0.5
1
Control Study A:
critical state from arbitrary states
Control Study B:
z-polarized state from PM ground states
Control Study C:
critical state from PM ground states

Figure 5.2: Many-body control studies in the ground state phase diagram of the quantum Ising model, analyzed in this work: an RL agent is trained to prepare the critical state of the transverse field Ising model from random initial states (Ctrl Study A, magenta), the z-polarized product state from a class of paramagnetic ground states (Ctrl Study B, green), and the critical state of the mixed field Ising model from paramagnetic ground states of opposite interaction strength (Ctrl Study C, cyan).

of interacting quantum particles, as required by present-day quantum simulators. Our proposed QMPS framework is illustrated in Fig. 5.1.

As a concrete example, we consider the problem of state preparation and present three case studies in which we prepare different ground states of the paradigmatic mixed field Ising chain (see Fig. 5.2). We train QMPS agents to prepare target states from a class of initial (ground) states, and devise universal controls with respect to experimentally relevant sets of initial states. In contrast to conventional quantum control algorithms (such as CRAB or GRAPE [366, 367, 374]), once the optimization is complete, RL agents retain information during the training process in form of a policy or a value function. When enhanced with a deep learning architecture, the learned control policy generalizes to states not seen during training. We demonstrate how this singular feature of deep RL allows our agents to efficiently control quantum Ising chains (i) starting from various initial states that the RL agent has never encountered, and (ii) in the presence of faulty or noisy controls and stochastic dynamics. Thus, even in analytically intractable many-body regimes, an online RL agent produces particularly robust control protocols.

This chapter is organized as follows: In Section 5.2, I introduce the problem of quantum many-body control. In Section 5.3, I revisit the Q-learning algorithm and explain how it can be enhanced with function approximation for continuous state spaces. Section 5.4 establishes the reinforcement learning problem and our proposed framework, that is, a Q-learning agent based on matrix product states. The results of applying our scheme to different state preparation scenarios are presented in Section 5.5. In Section 5.6 I discuss how the QMPS framework can be intregatd in present-day NISQ device simulations. Section 5.7 provides additional details regarding the control studies. And finally, in Section 5.8 I present the details of the QMPS optimization.

5.2 Quantum many-body control

Consider a quantum many-body system in the initial state $|\psi_i\rangle$. Our objective is to find optimal protocols that evolve the system into a desired target state $|\psi_*\rangle$. We construct these protocols as a sequence of q consecutive unitary operators $U(\tau) = \prod_{j=1}^{q} U_{\tau_j}$, where $U_{\tau_j} \in \mathcal{A}$ are chosen from a set $\mathcal{A}$. To assess the quality of a given protocol, we compute the fidelity of the evolved state w.r.t. the target state:

$$F(\tau) = |\,\langle\psi_*|\,U(\tau)\,|\psi_i\rangle\,|^2. \tag{5.1}$$

Throughout the study, we focus on spin-1/2 chains of size N with open boundary conditions. The system on lattice site j is described using the Pauli matrices X_j, Y_j, Z_j. As initial and target states we select area-law states, e.g., ground states of the quantum Ising model (see Section 5.5). In order to control chains composed of many interacting spins, we obtain the target ground state using DMRG [36, 126], and represent the quantum state as an MPS throughout the entire time evolution.

We choose a set of experimentally relevant control unitaries $\mathcal{A}$ which contains uniform nearest-neighbor spin-spin interactions, and global rotations: $\mathcal{A} = \{\mathrm{e}^{\pm i\delta t_\pm \hat{A}_j}\}$, with

$$\begin{aligned}\hat{A}_j \in \mathcal{A} \quad = \quad & \Big\{ \sum_i \hat{X}_i, \sum_i \hat{Y}_i, \sum_i \hat{Z}_i, \\ & \sum_i \hat{X}_i\hat{X}_{i+1}, \sum_i \hat{Y}_i\hat{Y}_{i+1}, \sum_i \hat{Z}_i\hat{Z}_{i+1} \Big\}.\end{aligned} \tag{5.2}$$

Two-qubit unitaries are capable of creating/decreasing the entanglement of the state. Note that MPS-based time evolution is particularly efficient for such locally applied operators and the resulting protocols can be considered as a series of quantum gates.

The time duration (or angle) $\delta t_\pm$ of all unitary operators is fixed and slightly different in magnitude for positive and negative generators $\hat{A}_j$, and kept constant throughout the time evolution. Hence, the problem of finding an optimal sequence reduces to a discrete combinatorial optimization in the high-dimensional space of all possible series. For a fixed sequence length q, the number of all distinct sequences is $|\mathcal{A}|^q$; therefore, a brute-force search quickly becomes infeasible and more sophisticated algorithms, such as RL, are needed. By fixing both q and $\delta t_\pm$ prior to the optimization, in general, we will not be able to come arbitrarily close to the target state.

5.3 Background

5.3.1 Tabular Q-learning

In RL, a control problem is defined within the framework of an environment that encompasses the physical system to be controlled, and a trainable agent which chooses control actions to be applied to the system (see Fig. 5.3) [86]. The environment is described by a state space $\mathcal{S}$ and a set of physical laws that govern the dynamics of the system. At each time step t during the episode, the agent observes the current state

$s_t \in \mathcal{S}$ and receives a scalar reward signal r_t. The agent then chooses actions according to a strategy, called policy $\pi(a|s)$ – a function that assigns a probability to every action a depending on the current state s of the environment [86]. The goal is to find the optimal policy $\pi^*(a|s)$, i.e., the optimal action to take in any state s that maximizes the expected return $R = \mathbb{E}_\pi \left[\sum_{t=0}^{T} r_t | s_0 = s\right]$ starting from state s and following the policy π. In this work, we consider episodic tasks involving a termination condition (e.g., a fidelity threshold) which the agent has to reach within a fixed number of steps T. Once the termination condition is satisfied, the episode is over and the environment is reset. This is in contrast to non-episodic tasks which continue indefinitely.

Q-learning is a model-free RL algorithm in which the agent learns an optimal policy $\pi^*(a|s)$ solely via observing environment transitions, i.e., without knowing or building a representation of the environment dynamics, and without access to any prior information about the system [351]. To every fixed policy π (optimal or sub-optimal), we can assign a Q-function, defined as the expected return starting from state s, taking action a, and following the policy π afterwards:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, a_0 = a\right]. \tag{5.3}$$

The discount factor $\gamma \in (0,1]$ gives a higher importance to immediate rewards and therefore ensures stability for continuing, non-episodic RL tasks.

In Q-learning, the optimal policy π^* is found indirectly through learning the optimal Q-value function $Q^*(s,a)$ that gives the maximum expected cumulative discounted reward:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a). \tag{5.4}$$

Once the optimal Q-values are known, the optimal policy is deterministic: $\pi^*(s) = \arg\max_a Q^*(s,a)$, i.e., it is given by greedily taking actions according to the maximum optimal Q-value in each state.

When the state space is discrete, the optimal Q-function can be learned using tabular Q-learning through an iterative update rule derived from the Bellman optimality equation [351]

$$\begin{aligned} Q_{k+1}(s,a) &\leftarrow Q_k(s,a) + \alpha\delta_k, \\ \delta_k &= r(s,a) + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a), \end{aligned} \tag{5.5}$$

where k denotes the iteration step of the algorithm, $\alpha \in (0,1]$ is the learning rate, and δ_k is the temporal difference error. Note that Q-learning requires isolated tuples (s,a,r,s'), known as transitions, and not complete trajectories. Moreover, due to the presence of the max function in the update-rule above, the algorithm is *off-policy*: this means that the transitions can come from any policy (also old ones) – and yet the new updated Q-function approaches the optimal Q^*.

Convergence is guaranteed if each possible state-action pair (s,a) can, in principle, be visited infinitely often. To fulfill this condition the agent has to *explore* sufficiently different state-action pairs. At the same time, the agent should also *exploit* the high-reward transitions, especially towards the end of training when the Q-value estimates have mostly converged to their true values. A common choice of behavior policy to follow during training that satisfies this Exploration-Exploitation dilemma, is an ϵ-greedy policy:

$$a = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_{a'} Q^{\pi}(s,a') & \text{otherwise} \end{cases}, \tag{5.6}$$

i.e. the agent chooses a random action with some small probability ϵ and the greedy action maximizing the Q-value otherwise. The hyperparameter ϵ can be decreased, e.g., exponentially, starting from a value close to 1 (exploration-dominated regime) at the beginning of training, to a small value, e.g., $\epsilon = 0.01$, leading to less exploration and more exploitation as training progresses.

5.3.2 Deep Q-Learning

For large or continuous state spaces, such as Hilbert spaces, the tabular Q-learning algorithm described above is inapplicable. In such cases, it is only possible to learn an approximation to the optimal Q-values, $Q_\theta(s,a) \approx Q^*(s,a)$, given by a parameterized function, e.g., a neural network [87]. The parameters θ of the variational ansatz are then optimized by minimizing the expected mean-square temporal difference error

$$\begin{aligned} C_k(\theta_k) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{R}}\left[\Big(y_k - Q_{\theta_k}(s,a)\Big)^2\right], \\ y_k &= r + \gamma \max_{a'} Q_{\bar{\theta}_k}(s',a')\,. \end{aligned} \tag{5.7}$$

The minibatch of transitions (s,a,r,s'), used in each optimization step k, is uniformly sampled from a fixed-size replay buffer $\mathcal{R}$ that contains previously collected transitions from agent-environment interactions. Since Q-learning is an off-policy algorithm, transitions used for updating the Q-value do not have to coincide with the target policy allowing the use of experience replay. Thus, the subroutine of collecting environment transitions can be run independently and, if necessary, in parallel to the optimization subroutine, thus speeding up training. Therefore, the use of a replay buffer makes Q-learning more data-efficient than policy gradient methods.

Note that the RL loss function C_k in Eq. (5.7) is different from the loss in supervised learning, in that the regression target $y_k = r + \gamma \max_{a'} Q_{\bar{\theta}_k}(s',a')$ itself depends on the parameterized Q-values that have to be learned; therefore, the target (i.e., the label) changes in the course of training. This running target makes DQN different from ordinary gradient descent, and is the reason for the lack of convergence guarantees in DQN. To stabilize deep Q-learning, a second *target* Q-value network $Q_{\bar{\theta}_k}(s,a)$ is introduced whose parameters $\bar{\theta}$ are held fixed during the optimization step. The optimized parameters θ are periodically copied to the target network $\bar{\theta} \leftarrow \theta$.

Finally, we also employ Double Q-learning to reduce overestimation errors in the Q-values [375]. Here, the regression target is replaced by

$$y_k^{\text{Double}} = r + \gamma\ Q_{\bar{\theta}_k}(s', \underset{a'}{\text{argmax}}\ Q_{\theta_k}(s',a')). \tag{5.8}$$

5.4 Proposal

5.4.1 Reinforcement learning (RL) framework

In the following, we outline the control scenario to be solved by the RL agent, i.e., we define the RL states, actions, and the reward function.

States — In our quantum many-body control setting, the RL state space $\mathcal{S}$ comprises all quantum states $|\psi\rangle$ of the 2^N-dimensional many-body Hilbert space. Here, we consider states in the form of an MPS with a fixed bond dimension χ_ψ: if $\chi_\psi < \chi_{\max}$ is smaller than the maximum bond dimension $\chi_{\max} = 2^{N/2}$, long-range entanglement cannot be fully captured, and the resulting MPS becomes a controlled approximation to the true quantum state.Hence, state preparation of volume-law entangled states is restricted to intermediate system sizes when using MPS. On the other hand, for large system sizes, the control problems of interest typically involve initial and target states that are only weakly entangled such as ground states of local many-body Hamiltonians. In these cases, the optimal protocol may not create excessive entanglement suggesting that the system follows the ground state of a family of local effective Hamiltonians, similar to shortcuts-to-adiabaticity control [376], and thus, justifying a MPS-based description.

Actions — If not specified otherwise, the set of available actions $\mathcal{A}$ contains local spin-spin interactions and single-particle rotations, as defined in Eq. (5.2).

Rewards — Since our goal is to prepare a specific target state, a natural figure of merit to maximize is the fidelity $F_t = |\langle\psi_t|\psi_*\rangle|^2$ between the current state $|\psi_t\rangle$ and the target state $|\psi_*\rangle$. To avoid a sparse-reward problem caused by exponentially small overlaps in many-body systems, we choose the log-fidelity per spin at each time step as a reward: $r_t = N^{-1}\log(F_t)$. Moreover, we set a fidelity threshold F^*, which the agent has to reach for an episode to be terminated successfully. Note that the agent receives a negative reward at each step; this provides an incentive to reach the fidelity threshold in as few steps as possible, in order to avoid accruing a large negative return $R = \sum_{t=1}^{T} r_t$, thus leading to short optimal protocols. For assessing the performance of the QMPS agent to prepare the target state, we show the final single-particle fidelity $F_{\text{sp}} = \sqrt[N]{F}$ as it represents a more intuitive quantity than the related log fidelity used in quantum simulation experiments.

We note in passing that we do not fix the length of an episode (the number of protocol steps) beforehand and the agent is always trying to find the shortest possible protocol to prepare the target state. However, we terminate each episode after a maximum number of allowed steps even if the target state has not been successfully prepared yet: otherwise episodes, especially at the beginning of training, can become exceedingly long leading to unfeasible training times.

5.4.2 Matrix product state ansatz for Q-learning (QMPS)

We choose Q-learning to train our RL agent (see Section 5.3.1), since it is off-policy and, thus, more data-efficient compared to policy gradient methods. The optimal Q-function $Q^*(\psi, a)$ defines the total expected return starting from the state $|\psi\rangle$, selecting the action a and then following the optimal protocol afterwards. Intuitively, the optimal

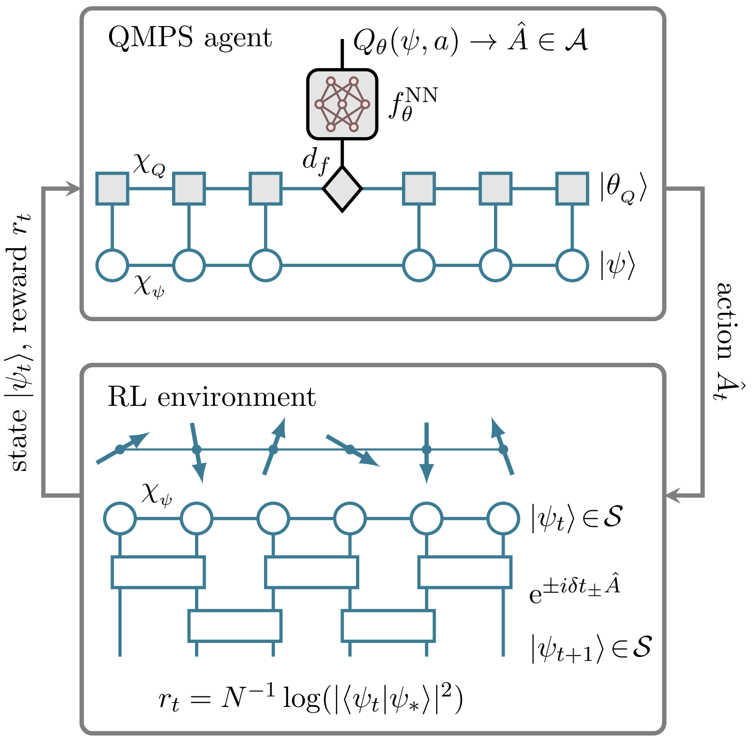

QMPS agent
$Q_\theta(\psi, a) \to \hat{A} \in \mathcal{A}$
f_θ^{NN}
d_f
χ_Q
$|\theta_Q\rangle$
χ_ψ
$|\psi\rangle$
state $|\psi_t\rangle$, reward r_t
action $\hat{A}_t$
RL environment
χ_ψ
$|\psi_t\rangle \in \mathcal{S}$
$\mathrm{e}^{\pm i\delta t_\pm \hat{A}}$
$|\psi_{t+1}\rangle \in \mathcal{S}$
$r_t = N^{-1}\log(|\langle\psi_t|\psi_*\rangle|^2)$

Figure 5.3: Q-learning framework (QMPS) based on matrix product states. The RL environment encompasses a quantum many-body spin chain represented in compressed MPS form which is time evolved according to globally applied unitary operators chosen from a predefined set $\mathcal{A}$. The reward r_t is given by the normalized log-fidelity between the current state $|\psi_t\rangle$ and the target $|\psi_*\rangle$. The QMPS agent is represented by a parameterized Q-value function $Q_\theta(\psi, a)$ composed of a MPS $|\theta_Q\rangle$ which is contracted with the quantum state MPS $|\psi_t\rangle$, and a subsequent neural network (NN) which outputs a Q-value for each different action $\hat{A}$. The trainable parameters of the QMPS are determined by the feature vector dimension d_f and the bond dimension χ_Q.

action in a given state maximizes $Q^*(\psi, a)$. Hence, if we know $Q^*(\psi, a)$ for every state-action pair, we can solve the control task. In Q-learning this is achieved indirectly, by first finding Q^*. This approach offers the advantage to re-use the information stored in Q^* even after training is complete.

Since the state space is continuous, it becomes infeasible to learn the exact Q^*-values for each state. Therefore, we approximate $Q^* \approx Q^*_\theta$ using a function parametrized by variational parameters θ, and employ the DQN algorithm to train the RL agent [87]. In this work, we introduce a novel architecture for the Q^*-function, based on a combination of a MPS and a NN, called QMPS, which is specifically tailored for quantum many-body states that can be expressed as a MPS (see Fig. 5.3). We emphasize that the QMPS is independent of the MPS representation of the quantum state, and has its own bond dimension χ_Q.

To calculate $Q_\theta(\psi, a)$ for each possible action a in a quantum state $|\psi\rangle$, we first compute the overlap between the quantum state MPS and the QMPS. The contraction of two MPS can be performed efficiently and scales only linearly in the system size for fixed bond dimensions. The output vector of the contraction corresponding to the dangling leg of the central QMPS tensor, is then interpreted as a feature vector of dimension d_f, which is used as an input to a small fully-connected neural network (see Fig. 5.3). Adding a NN additionally enhances the expressivity of the Q^*_θ ansatz by making it nonlinear. The final NN output contains the Q^*-values for each different action.

The QMPS feature vector can be naturally written as an overlap between the quantum state MPS $|\psi\rangle$ and the QMPS $|\theta_Q\rangle$. Thus, the Q^*-value can be expressed as

$$Q_\theta(\psi, a) = f_\theta\left(N^{-1}\log\left(|\langle\theta_Q|\psi\rangle|^2\right)\right), \tag{5.9}$$

where $f_\theta(\cdot)$ denotes the neural network. We additionally apply the logarithm and divide by the number of spins N in order to scale the QMPS framework to a larger number of particles. Note also that the QMPS does not represent a physical wave function (it is not normalized); however, for ease of notation, we still express it using the bra-ket formalism.

Thus, the trainable parameters θ of the Q^*-function contain the N+1 complex-valued QMPS tensors $|\theta_Q\rangle$, plus the real-valued weights and biases of the subsequent NN. The QMPS feature dimension d_f and the QMPS bond dimension χ_Q are hyperparameters of the optimization, which determine the number of variational parameters of the MPS in analogy to the hidden dimension of neural networks.

Note that the resources (time and memory) for training the QMPS framework scale at worst polynomially in any of the parameters of the system and the ansatz, such as the QMPS bond dimension χ_Q, the feature dimension d_f, and the local Hilbert space dimension $d=2$. Furthermore, QMPS reduces an exponential scaling of the resources with the system size N to a linear scaling in N, therefore, allowing efficient training on large spin systems.

5.5 Results

Our MPS-based RL framework is specifically designed for preparing low-entangled states in 1d, such as ground states of local gapped Hamiltonians. Hence, in the subsequent case studies we consider ground states of the 1d mixed field Ising model as an exemplary system:

$$\hat{H}_{\text{Ising}} = J\sum_{j=1}^{N-1}\hat{Z}_j\hat{Z}_{j+1} - g_x\sum_{j=1}^{N}\hat{X}_j - g_z\sum_{j=1}^{N}\hat{Z}_j, \tag{5.10}$$

where g_x (g_z) denotes a transverse (longitudinal) field. In the case of negative interaction strength and in the absence of a longitudinal field $g_z = 0$, the system is integrable, and has a critical point at $g_x = 1$ in the thermodynamic limit, separating a paramagnetic (PM) from a ferromagnetic phase (FM) (see Fig. 5.2). For $g_z > 0$, the model has no known closed-form expressions for its eigenstates and eigenenergies. In addition, for positive interactions, the phase diagram features a critical line from $(g_x, g_z) = (1, 0)$ to $(g_x, g_z) = (0, 2)$ exhibiting a transition from a paramagnetic to an antiferromagnetic phase (AFM).

5.5.1 Universal ground state preparation from arbitrary initial quantum states for $N = 4$ spins

In the noninteracting limit, $J = 0$, the QMPS agent readily learns how to control a large number of spins (see Section 5.7.1). Instead, as a nontrivial benchmark of the QMPS framework, here we teach an agent to prepare the ground state of the 4-spin transverse field Ising model at $(J = -1, g_x = 1, g_z = 0)$, starting from *randomly drawn* initial states. While this control setup can be solved using the full wave function and a conventional neural network ansatz, the uniform initial state distribution over the entire continuous Hilbert space creates a highly non-trivial learning problem and presents a first benchmark for our QMPS framework. Moreover, system sizes of $N \sim 4$ spins already fall within the relevant regime of most present-day studies using quantum computers, where gate errors and decoherence currently prevent exact simulations at larger scales [167, 285, 377].

We first train an agent (QMPS-1) to prepare the target ground state within 50 protocol steps or less, setting a many-body fidelity threshold of $F^* \approx 0.85$. The initial states during training are chosen to be (with probability $p = 0.25$) random polarized product states, or (with probability $p = 0.75$) random reflection-symmetric states drawn from the full $2^4 = 16$ dimensional Hilbert space[*]. In this way the QMPS-1 agent has to learn to both disentangle highly entangled states to prepare the Ising ground state, but also to appropriately entangle product states to reach the entangled target (the learning curves of the QMPS-1 agent are shown in Fig. 5.12 in Section 5.7.1). After this training stage, we test the QMPS-1 agent on a set of 10^3 random initial states and find that in $\sim 99.8\%$ of the cases the fidelity threshold is successfully reached within the 50 allowed steps. A (much) better fidelity cannot be achieved by the QMPS-1 agent alone, due to the discreteness of the action space and the constant step size used, rather than limitations intrinsic to the algorithm.

To improve the fidelity between the final and the target state, we now train a second, independent agent (QMPS-2) with a tighter many-body fidelity threshold of $F^* \approx 0.97$. The initial states are again sampled randomly as mentioned above; however, we first use the already optimized QMPS-1 agent to reach the vicinity of the target state within $F > 0.85$. Then, we take those as initial states for the training of the second QMPS-2 agent. This two-stage learning schedule can in principle be continued to increase the fidelity threshold even further. The learning curves of the QMPS-2 optimization are shown in Fig. 5.4(a). In Fig. 5.4(b)-(c) we present the obtained protocols for four exemplary initial states. Overall, the combined two-agent QMPS is able to reach the fidelity threshold of $F^* \sim 0.97$ for $\sim 93\%$ of the randomly drawn initial states within the 50 episode steps that were imposed during training. We emphasize that this result is already nontrivial, given the restricted discrete action space, and the arbitrariness of the initial state.

Let us now exhibit two major advantages of RL against conventional quantum control algorithms. (i) After training we can double the allowed episode length for each agent to 100 steps. Since this allows for longer protocols, we find that the target

[*]Due to the enforced reflection symmetry, the state space we effectively sample from is 10 dimensional. We sample (Haar) random states by drawing the wave function amplitudes from a normal distribution followed by normalization.

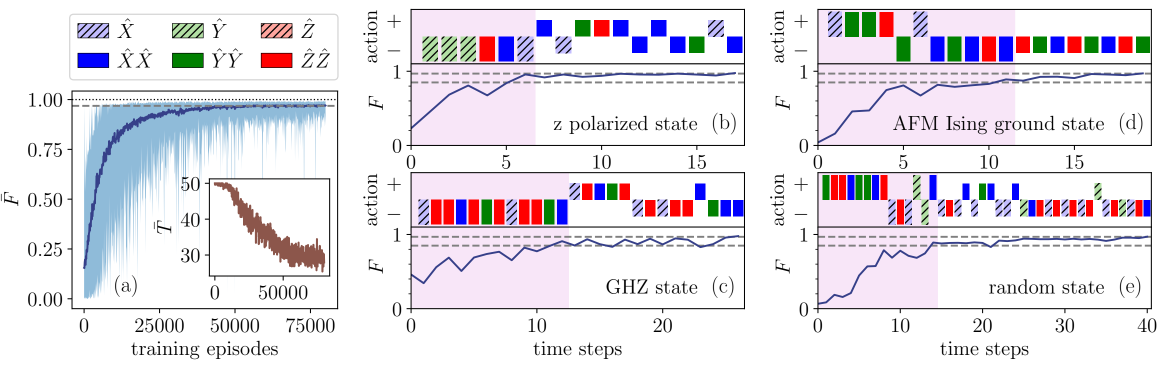


Figure 5.4: Universal four-qubit control. (a) Achieved many-body fidelity $\bar{F}$ between final and target state during training averaged over 100 training episodes (dark-blue curve). The best and worst fidelity within each episode window is indicated by the light-blue-shaded area. The fidelity threshold, $F^* \sim 0.97$, is marked by a gray dashed line. The inset shows the mean number of episode steps $\bar{T}$ during training (averaged over 100 episodes). The maximum number of allowed steps is set to 50. **(b)-(e)** Two QMPS agents (see text) are trained with fidelity thresholds $F^* \sim 0.85, 0.97$ (gray dashed lines), to prepare the Ising ground state $(J = -1, g_x = 1, g_z = 0)$, starting from (b) the z-polarized product state, (c) the GHZ state, (d) an Ising antiferromagnetic ground state at $(J = +1, g_x = g_z = 0.1)$, and (e) a Haar-random state. The QMPS-2 agent starts from the final state reached by the QMPS-1 agent (purple shaded area). The many-body fidelity F between the instantaneous and the target state, is shown in the lower part of each panel. The upper part of the panels shows the control protocol. The colors and shading of each rectangle indicate the applied action A (see legend); $\pm$ stands for the sign of the action generator, i.e., $\exp(\pm i\delta t_\pm \hat{A})$. The QMPS-1,2 agents use fixed time steps of $\delta t_\pm = (\pi/8, \pi/16)_+, (\pi/13, \pi/21)_-$, indicated by action rectangles of different sizes in the protocol. $N = 4$ spins.

state can be successfully prepared for 99.5% of the initial states (compared to the previously observed 93%). Note that this feature is a unique advantage of (deep) RL methods, where the policy depends explicitly on the quantum state: during training, the agent learns how to take optimal actions starting from *any* quantum state and hence, it is able to prepare the target state if it is given sufficient time. Moreover, (ii) in this example we achieve universal quantum state preparation, i.e., the trained RL agent succeeds in preparing the target state irrespective of the initial state. This is not possible with conventional control techniques where the optimal protocol is usually tailored to a specific initial state, and the optimization has to be rerun when starting from a different state.

5.5.2 Preparation of a polarized product state from paramagnetic ground states for $N = 32$ spins

In general, a Haar-random quantum many-body state is volume-law entangled and, hence, it cannot be approximated by a MPS of a fixed bond dimension. Moreover, it becomes increasingly difficult to disentangle an arbitrarily high entangled state for

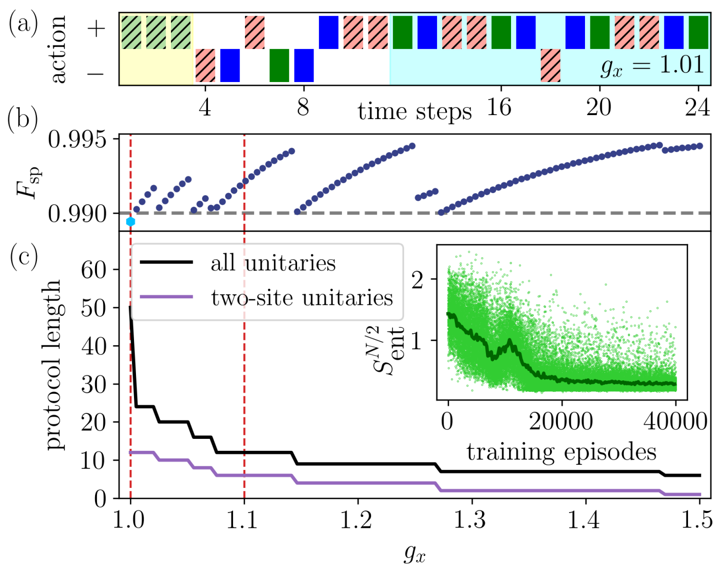


Figure 5.5: Transverse-field Ising control. (a) Optimal protocol obtained, starting from an initial ground state at $g_x = 1.01$ (see Fig. 5.4 for action legend). The cyan shaded segment indicates a generalized Euler-angle-like many-body rotation. **(b)** Final single-particle fidelities $F_{\rm sp}$ starting from initial ground states with transverse field value g_x. The target state is the z-polarized product state. The gray dashed line denotes the fidelity threshold: it is surpassed for most initial states except at the critical point $g_x \sim 1$ (cyan dot). The red vertical dashed lines contain the training region. **(c)** The number of actions (unitaries) in the QMPS protocols versus the initial state parameter g_x. The protocol starting from the critical state ($g_x \sim 1$) does not reach the fidelity threshold and is truncated after 50 episode steps. Inset: the half-chain von Neumann entanglement entropy of final states during training decreases as learning improves. The dark green curve denotes the average over 200 episodes. $N = 32$ spins.

larger system sizes [378]. Therefore, when working in the truly quantum many-body regime, we have to restrict to initial and target states that are not volume-law entangled. As an example, here we consider a many-body system of $N = 32$ spins and learn to prepare the z-polarized state from a class of transverse field Ising ground states ($J = -1, g_z = 0$). Once high-fidelity protocols are found, they can be inverted to prepare any such Ising ground state from the z-polarized state, which presents a relevant experimental situation (see Section 5.7.2). Many-body ground state preparation is a prerequisite for both analog and digital quantum simulation, and enables the study of a variety of many-body phenomena such as the properties of equilibrium and nonequilibrium quantum phases and phase transitions.

To train the agent, we randomly sample initial ground states on the paramagnetic side of the critical point: $1.0 < g_x < 1.1$. The difficulty in this state preparation task is determined by the parameter g_x defining the initial state: states deeper into the paramagnetic phase are more easy to 'rotate' into the product target state, while states close to the critical regime require the agent to learn how to fully disentangle the initial state in order to reach the target. We train a QMPS agent on a system of $N = 32$ spins which is infeasible to simulate using the full wavefunction, and is far out-of-reach for neural-network based approaches. We set the single-particle fidelity threshold to $F^*_{\mathrm{sp}} = 0.99$ and allow at most 50 steps per protocol.

Figure 5.5(b) shows the successfully reached final fidelity when the trained QMPS agent is tested on unseen initial states, for various values of g_x. First, notice that the agent is able to prepare the target state also for initial states with $g_x > 1.1$ that lie outside of the training region (dashed vertical lines). Hence, we are able to extrapolate optimal control protocols well beyond the training data distribution, without additional training. However, this is not true for states inside the critical region, $g_x \lesssim 1$, and in the ferromagnetic phase ($g_x \ll 1$); such a behavior is not surprising, since these many-body states have very different properties compared to those used for training. Interestingly, it follows that the onset of criticality can be detected in the structure of control protocols, as the number of required gates (actions) and, in particular, of entangling unitaries, increases rapidly as one approaches the critical point (see Fig. 5.5(c)).

Discontinuities in the achieved fidelity as can be seen in Fig. 5.5(b) arise due to the fixed, constant step size: we observe distinct jumps in the final fidelity, whenever the length of the protocol sequence increases. This is a primary consequence of the discrete control action space. Its physical origin can be traced back to the need for a more frequent use of disentangling two-site unitaries, for initial states approaching the critical region.

Figure 5.5(a) shows the optimal protocol at $g_x = 1.01$: first, the agent concatenates three $\hat{Y}$-rotations ($\delta t_+ = \pi/12$) in a global gate, which shows that it learns the orientations of the initial x-paramagnet and the z-polarized target (yellow shaded region). This is succeeded by a non-trivial sequence containing two-body operators. A closer inspection (see Fig. 5.16 in Section 5.7.2) reveals that the agent discovered a generalization of Euler-angle rotations in the multi-qubit Hilbert space (blue shaded region). This is remarkable, since it points to the ability of the agent to construct compound rotations, which is a highly non-trivial combinatorial problem for experimentally relevant constrained action spaces. This can be interpreted as a generalization of dynamical decoupling sequences used in state-of-the-art NMR experiments. We verified that this protocol is a local minimum of the control landscape.

We also investigated the system size dependence of optimal protocols in this control study. To our surprise, we find that agents trained on the $N = 32$ spin system produce optimal protocols that perform reasonably well on smaller ($N = 8$) as well as larger ($N = 64$) systems. Hence, this control problem admits a certain degree of transferability, which worsens for initial states closer to the finite-size dominated critical region (see Section 5.7.2).

The MPS-based control framework enables us to readily analyze the physical entanglement growth during training, via the bond dimension of the quantum state χ_ψ. The protocol exploration mechanism in QMPS causes the agent to act mostly randomly during the initial stages of learning. This translates to random sequences of unitary gates that can lead to an increase of entanglement entropy (see inset of Fig. 5.5(c)). In our simulations, we set the maximum allowed bond dimension to $\chi_\psi = 16$, which is sufficient for the considered initial and target states to be approximated reliably. However, not all states encountered can be represented with such a small bond dimension, as reflected by large truncation errors during training (see Fig. 5.18 in Section 5.7.2). Nonetheless, as training progresses, the agent learns to take actions that do not create excessive entanglement as is shown in Fig. 5.5(c). Therefore, the truncation error naturally decreases, as training nears completion. As a consequence, the final converged protocols visit states that lie within a manifold of low-entangled states. Moreover, increasing χ_ψ does not change these results. We believe that this mechanism relies on the area-law nature of the initial and target states, and we expect it to open up the door towards future control studies deeper in the genuine many-body regime.

5.5.3 Learning robust critical-region state preparation for $N = 16$ spins

States in the critical region possess non-trivial correlations and show strong system-size dependence, which make manipulating them highly non-trivial. In particular, the required time duration to adiabatically prepare critical states diverges with the number of particles, whereas sweeping through critical points reveals properties of their universality classes [379]. Therefore, finding optimal control strategies away from the adiabatic limit is an important challenge. Critical state preparation is also of practical relevance for modern quantum metrology, where the enhanced sensitivity of critical states to external fields is leveraged to perform more precise measurements [361].

Our final objective is to prepare a ground state in the critical region of the mixed field Ising chain ($J = +1, g_x = 0.5, g_z = 1.5$) starting from non-critical paramagnetic ground states of the same model with flipped interaction strength: $J=-1$, $1.0<g_x<1.5$, $0<g_z<0.5$ (see Fig. 5.2). Hence, the agent has to learn to connect ground states of two distinct Hamiltonians. This scenario is often relevant in typical experimental setups where only a single-sign interaction strength can be realized: e.g., the initial state comes from the $J<0$ Ising model, while the ground state of interest belongs to the antiferromagnatic Ising Hamiltonian. In general, one can use two completely distinct parent Hamiltonians for the initial and target states, one of which being inaccessible in the quantum simulator platform at hand, while the other being the object of interest.

We train our QMPS agent on $N=16$ spins with a single-particle fidelity threshold of $F^*_{\mathrm{sp}}=0.97$, and a maximum episode length of 50. Figure 5.6(a) shows the achieved fidelity between the critical target state and the final state, for different initial ground states corresponding to a rectangular region in the (g_x, g_z)-plane. Notice that the agent is able to generalize to unseen initial states lying far outside the training region (white rectangle), and fails only close to the critical point of the transverse field Ising model ($g_x=1, g_z=0$) and for a few isolated initial states well outside of the training region.

We now demonstrate that our QMPS agent shows remarkable generalization capabilities in noisy environments. In particular, we analyze how robust the trained QMPS agent is to stochastic perturbations in the time evolution of the state – a common problem in noisy intermediate-scale quantum (NISQ) computing devices [49]. In what follows, we consider two different sources of noise independently: (i) At each time step, with probability ϵ, a random action rather than the selected one is enforced. This type of noise mimics bit- or phase-flip errors, which occur in quantum computing. (ii) Gaussian random noise with zero mean and standard deviation σ, is added to the time duration $\delta t_\pm$ of each unitary operator; this can, for instance, result from imperfect controls in the experimental platform.

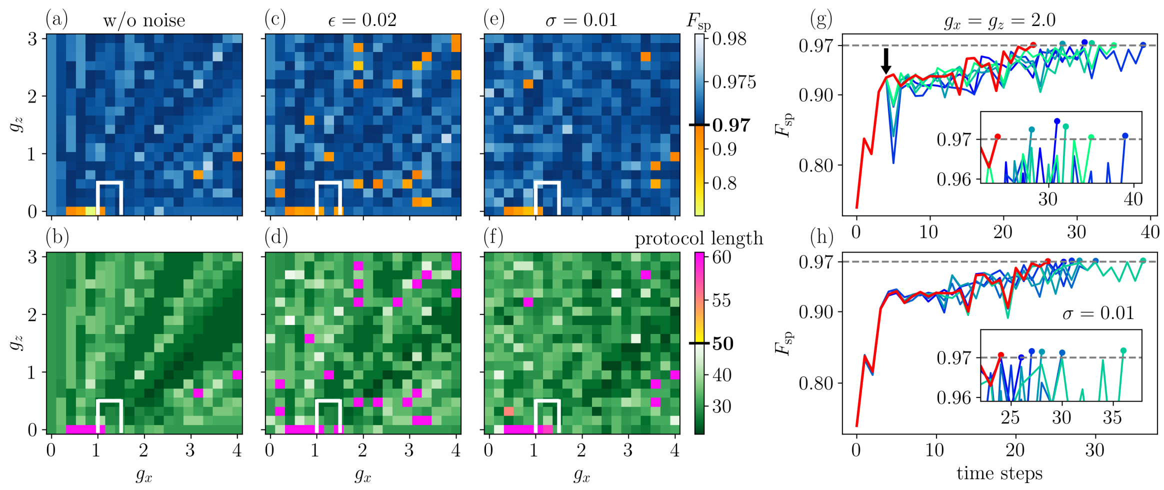


Figure 5.6: Self-correcting mixed-field Ising control. (a),(b) Final single-particle fidelity F_{sp} (top) and corresponding protocol length (bottom) – see colorbars – versus the initial Ising ground state parameter values g_x, g_z. The target is a critical state of the Ising model at $(J=+1, g_x=0.5, g_z=1.5)$. Training started only from initial states sampled randomly from the enclosed white rectangle. Each of the two parts of a colorbar is shown on a linear scale with the fidelity threshold ($F^*_{\mathrm{sp}}=0.97$) and the maximum episode length during training (50), indicated by short black lines. **(c)-(f)** Same as (a)-(b) but for noisy evolution – (c),(d): At each time step, actions other than the one selected by the agent, are taken with probability $\epsilon=0.02$; (e),(f): White Gaussian noise with standard deviation $\sigma=0.01$ is added to the time step $\delta t_\pm$ of all applied unitaries. **(g)** Time-dependence of the single-particle fidelity starting from an arbitrary initial ground state, and following the trained agent. The red curve denotes the unperturbed (noise-free) QMPS protocol. At time step 5 (indicated by the black arrow), the QMPS protocol is perturbed by enforcing 5 suboptimal actions. All subsequent actions are selected again according to the trained QMPS policy without perturbation (blue curves). The inset displays a zoom in the vicinity of the fidelity threshold (dashed gray line), showing that each protocol terminates successfully. **(h)** Same as in (g) but for dynamics subject to Gaussian noise at every time step $\delta t_\pm$, for 5 different random seeds giving rise to 5 distinct protocols. $N=16$ spins.

Noise type (i) is equivalent to using an ϵ-greedy policy. Hence, the states encountered when acting with such a policy, could have (in principle) been visited during training. Due to the generalization capabilities of RL, it is reasonable to expect that an agent will act optimally after non-optimal actions have occurred, attempting to correct the 'mistake'. In Fig. 5.6(c)-(d) we show the achieved final fidelity (top) and the required number of steps (bottom) for $\epsilon = 0.02$. Overall, the fidelity threshold can still be reached in the majority of test cases. The randomness typically results in longer protocols indicating that the agent indeed adapts to the new states encountered. Interestingly, in the noise-free case the agent fails to prepare the target state for a few points outside the training region (orange points in Fig. 5.6(a)); this can be attributed to incorrectly estimated Q-values which have not fully converged to the optimal ones outside of the training interval. However, when adding the perturbation, the agent is able to correct its mistake in one of the shown instances and prepares the target state successfully (see Fig. 5.6(c)).

Recall that we use a different time step $\delta t_\pm$ for positive and negative actions. This way the agent is not just able to undo a non-optimal action by performing its inverse; it rather has to adjust the entire sequence of incoming unitaries in a non-trivial way. The ability of the QMPS agent to adapt is demonstrated in Fig. 5.6(g) where we plot the fidelity during time evolution starting from an arbitrary initial ground state. At time step $t = 5$, we perturb the protocol by taking 6 different actions, and let the agent act according to the trained policy afterwards; this results in 6 distinct protocol branches. In each of them, the agent tries to maximize the fidelity and successfully reaches the fidelity threshold after a few extra protocol steps. In Section 5.7.3 we provide further examples showing that this behavior is generic, and can also be observed for different initial states.

In contrast to the ϵ-noise, adding Gaussian random noise σ, (ii), to the time step duration $\delta t_\pm$, results in states that the agent has not seen during training. This source of noise, therefore, explicitly tests the ability of the agent to generalize beyond the accessible state space, and in particular to interpolate between quantum many-body states. Fig. 5.6(e)-(f) displays the achieved fidelity and the corresponding protocol length for $\sigma = 0.01$. We find that the QMPS agent is also robust to this type of noise. In Fig. 5.6(h) we plot the fidelity trajectories starting from the same initial state using 5 different random seeds; this illustrates that our agent adapts successfully to previously unencountered many-body states, and steers the protocol on-line to reach beyond the fidelity threshold.

The robustness of QMPS agents to noise and, in general, to stochasticity in the quantum gates, demonstrates yet another advantage of deep RL methods over conventional quantum control techniques. The latter typically perform suboptimal in noisy systems since the optimization does not take into account the quantum state information during the time evolution, and the optimal protocols are specifically optimized for a fixed trajectory of quantum states [208]. By contrast, QMPS value functions are optimized on a large class of states and, as shown above, can interpolate and extrapolate to new, seen and unseen states as long as the deep learning approximation stays sufficiently accurate. Therefore, unlike conventional quantum control algorithms, QMPS agents have the ability to automatically self-correct their protocols on-the-fly, i.e., while the system is being time evolved.

5.6 NISQ device integration

The present QMPS framework requires the quantum state to be accessible at each time step for both training and inference purposes; yet, quantum states are not observable in experiments without performing expensive quantum state tomography. Nevertheless, there already exist efficient encoding strategies that map MPS into quantum circuits [380–384]. Moreover, several proposals were recently developed in which MPS are harnessed for quantum machine learning tasks, for example as part of hybrid classical-quantum algorithms [35, 135, 385] or as classical pre-training methods [386, 387]. Similar ideas can be applied to the QMPS architecture by mapping the trainable MPS

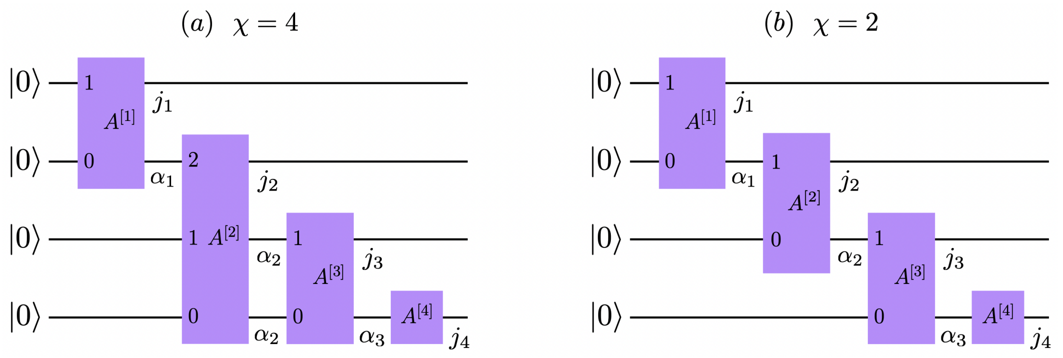


Figure 5.7: MPS to circuit mapping for the $N = 4$ MPS of Eq. (5.11) in left orthogonal form. (a) An MPS with bond dimensions $2-4-2$. (b) The truncated MPS with bond dimensions $2-2-2$.

to a parametrized quantum circuit, thus directly integrating the QMPS framework in quantum computations with NISQ devices. This would allow us to perform the expensive training routine on readily available classical computers while the inexpensive inference step can be performed on quantum hardware.

In the following we illustrate the QMPS to circuit mapping on the example of the universal ($N=4$) control study presented in Section 5.5.1. The QMPS state $\left|\theta_Q^\ell\right\rangle$ is represented as

$$\left|\theta_Q^\ell\right\rangle = \sum_{j_1,\dots,j_4}\sum_{\alpha_1,\alpha_2,\alpha_3} A^{[1]j_1}_{\alpha_1} A^{[2]j_2}_{\alpha_1\alpha_2} A^{[3]j_3;\ell}_{\alpha_2\alpha_3} A^{[4]j_4}_{\alpha_3} \left|j_1,\dots,j_4\right\rangle, \tag{5.11}$$

where we have already contracted the feature tensor with its neighboring tensor $A^{[3]}$ and ℓ denotes the feature vector index. Our goal is to rewrite the QMPS state as a quantum circuit $\left|\theta_Q^\ell\right\rangle = U_\theta^\ell\left|0\right\rangle$, where the state preparation unitary U_θ^ℓ is composed of several gates $U_\theta^\ell = G_4^\ell \cdots G_1^\ell$.

First, we transform the QMPS into the left canonical form via successive QR or singular value decompositions such that

$$\sum_{j_1\alpha_1} A^{[1]j_1}_{\alpha_1} A^{[1]j_1*}_{\alpha_1} = 1, \tag{5.12}$$

$$\sum_{j_i\alpha_i} A^{[i]j_i}_{\alpha_{i-1}\alpha_i} A^{[i]j_i*}_{\alpha'_{i-1}\alpha_i} = I_{\alpha_{i-1}\alpha'_{i-1}}, \qquad (i=2,3) \tag{5.13}$$

$$\sum_{j_4} A^{[4]j_4}_{\alpha_3} A^{[4]j_4*}_{\alpha'_3} = I_{\alpha_3\alpha'_3}. \tag{5.14}$$

In principle, the resultant tensors $A^{[i]}$ also depend on the feature index ℓ after performing the canonicalization. However, in what follows we omit the index ℓ and assume that all subsequent steps are performed for each of the indices separately.

The quantum circuit mapping of $\left|\theta_Q^\ell\right\rangle$ is depicted in Fig. 5.7(a). We can interpret the rightmost tensor $A^{[4]}$ as a single-qubit unitary, i.e., $G_4 = A^{[4]j_4}_{\alpha_3}$ as it satisfies Eq. (5.14). Similarly, we can rewrite the adjacent tensor $A^{[3]j_3}_{\alpha_2\alpha_3}$ with dimensions $4\times 2\times 2$ as a

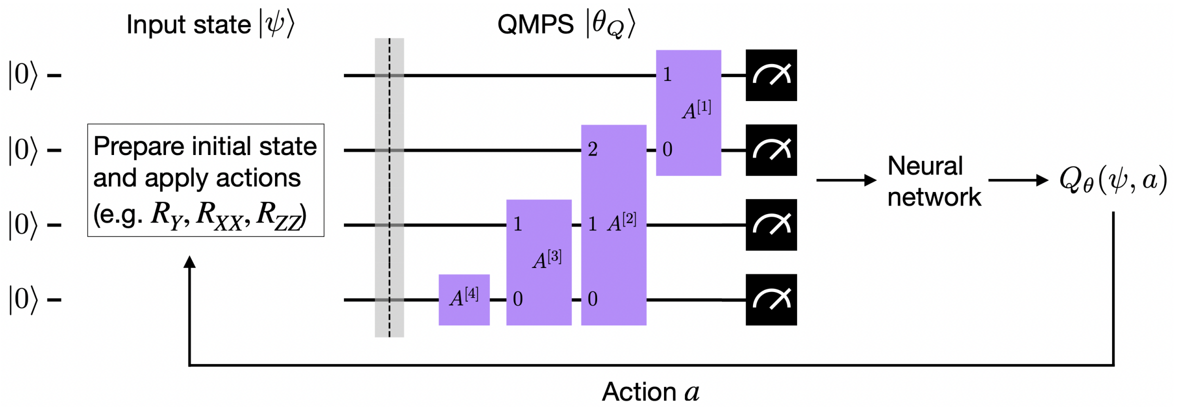


Figure 5.8: QMPS framework as a hybrid quantum-classical algorithm. On the quantum device, we first prepare the initial state and apply the already inferred protocol actions as gates. The resultant state $|\psi\rangle$ presents the input to the QMPS network. To compute the Q-values, we first apply the inverse of the QMPS circuit unitary $U_\theta^\dagger$ and measure the output in the computational basis. The rate of the all-zero measurement outcomes is an approximation to the fidelity $|\langle\theta_Q|\psi\rangle|^2$ and fed into the neural network on a classical computer. From the resultant Q-values we can infer the next action and repeat these steps until the target state is reached.

two-qubit unitary after reshaping the index α_2: $G_3 = A^{[3]j_3,\alpha_3}_{\alpha_2,\alpha_2'}$. The next tensor $A^{[2]j_2}_{\alpha_1\alpha_2}$ represents an isometry with input dimension 2 and output dimensions 4×2. Hence, we need to extend the columns of $A^{[2]}$ by padding it with the $(2^3 - 2)$ orthonormal vectors X in the kernel of $A^{[2]\dagger}$. The resultant square matrix $G_2 = [X \;\; A^{[2]}]$ is then chosen as the three-qubit unitary. Finally, we can apply the same steps to the remaining isometry $A^{[1]j_1}_{\alpha_1}$ which gives rise to the two-qubit gate G_1.

Using the above mapping, an MPS circuit with bond dimension $\chi = 2^n$ always contains at least one $(n+1)$-qubit gate. Thus, the $N = 4$ QMPS with bond dimension $\chi = 4$ results in a circuit including a three-qubit gate. However, the native gates realized in most present-day quantum computers contain at most two-qubit unitaries. Therefore, gates acting on more than two qubits first have to be decomposed into two- and single-qubit gates. Performing the decomposition in an exact manner is usually expensive, requires the use of optimization techniques, and often leads to very deep circuits nonetheless. With the short coherence times and large error rates of current quantum devices, it therefore quickly becomes infeasible to execute MPS circuits of bond dimension $\chi > 2$. Hence, we require alternative circuit mappings that give rise to at most two-qubit gates in the final circuit. The simplest approach is to truncate the given QMPS to a bond dimension $\chi = 2$ MPS (see Fig. 5.7(b) for the corresponding circuit). However, if the truncation errors are too large, the resultant circuit will not be an accurate description of the true quantum state anymore. Several approximative methods have been proposed to bridge this gap and prepare high fidelity states while restricting to the use of only two-qubit gates [380–384]. Note that all of these approaches can also be applied to the QMPS ansatz. For the remainder of this work we will consider the previously described exact mappings of the full $\chi = 4$ and truncated $\chi = 2$ QMPS as shown in Fig. 5.7.

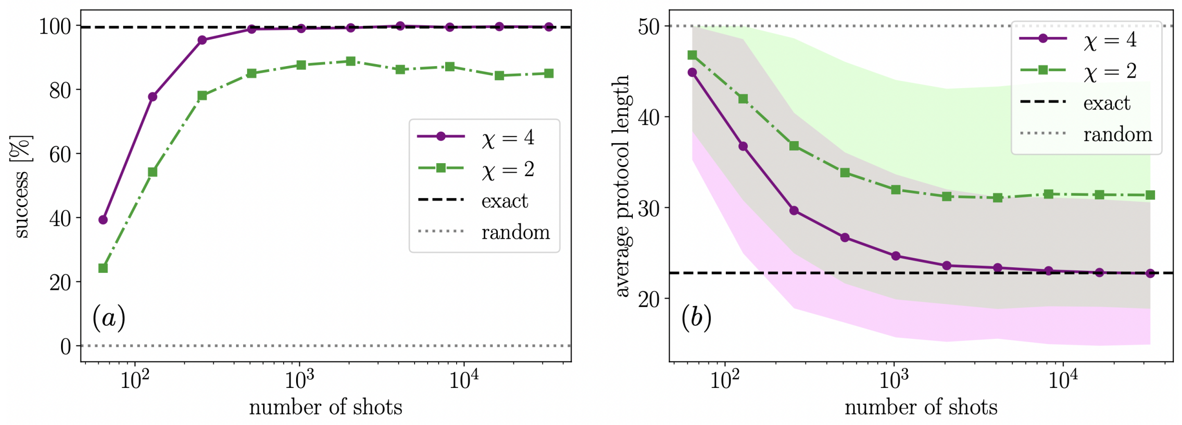


Figure 5.9: Universal four-qubit control. We sample 1000 random initial states and apply the QMPS circuit framework with a varying number of measurement shots. In (a) we display the percentage of runs in which the target state is successfully reached (i.e., the fidelity threshold of $F^* \sim 0.85$ is surpassed after at most 50 protocol steps). The success rate under exact computation (without sampling) is shown as a black dashed line. The success probability when acting completely random is zero (gray dotted line). We provide both, the results for the full $\chi = 4$ QMPS circuit (purple solid line) and the truncated $\chi = 2$ QMPS (green dash-dotted line). (b) The corresponding average number of required protocol steps for reaching the fidelity threshold. The standard deviation is indicated by the shaded areas. The black dashed line corresponds again to the average value computed via exact techniques, the gray dashed-dotted line indicates the maximum number of allowed episode steps (50).

To calculate the Q-values $Q_\theta(\psi, a)$ given an input quantum state $|\psi\rangle$, we first compute the fidelity between the input and the QMPS state

$$\left|\langle\theta_Q^\ell|\psi\rangle\right|^2 = \left|\langle 0|\, U_\theta^\dagger U_\psi \,|0\rangle\right|^2, \tag{5.15}$$

which can be obtained via sampling on a quantum computer. Alternatively, the overlap can also be accessed by performing a SWAP test, albeit requiring additional ancilla qubits and non-local gates [388, 389]. The computed fidelities for each QMPS circuit are then fed into the classical neural network giving rise to a hybrid quantum - classical machine learning architecture as shown in Fig. 5.8. If necessary, the parameters of the QMPS circuit U_θ can be finetuned by performing some additional optimization.

We test the QMPS circuit framework on the universal ground state preparation task of Section 5.5.1. In the following we only report results for the QMPS-1 agent trained on a fidelity threshold of $F^* \sim 0.85$. We translate the optimized QMPS to the corresponding quantum circuit and first investigate how the number of measurement shots for sampling the fidelity in Eq. (5.15) affects the performance of the QMPS protocols under an ideal (noise-free) simulator. To that end, we sample 1000 random initial states and compute the percentage of successfully prepared states (i.e. those runs for which the fidelity threshold is reached within 50 protocol steps). Moreover, we also store the corresponding average protocol length and show the results in Fig. 5.9(a)-(b). Surprisingly, as few as 500 shots are already sufficient to reach success rates of

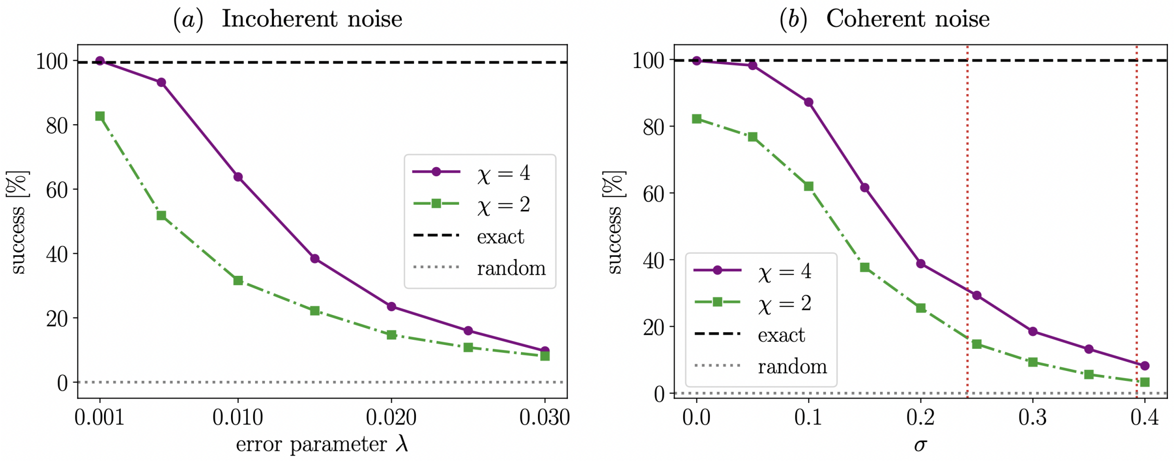


Figure 5.10: Universal four-qubit control. We sample 1000 random initial states and apply the QMPS circuit framework in the presence of noise. We set the number of measurement shots to 4096 and plot the percentage of runs in which the target state is successfully reached against the noise strength. The success rate under exact, noise-free computation (without sampling) is shown as a black dashed line. The success probability when acting completely random is zero (gray dotted line). We provide both, the results for the full $\chi = 4$ QMPS circuit (purple solid line) and the truncated $\chi = 2$ QMPS (green dash-dotted line). (a) The success rate when adding a depolarizing noise channel (see. Eq. (5.16)) with error parameter λ after each gate. Note that the noise parameter for all single-qubit gates is always fixed to $\lambda_1 = 10^{-4}$. (b) The success percentage when adding Gaussian random noise with standard deviation σ to the time step duration $\delta t_\pm$ of each action. For comparison, the original, unperturbed time step sizes $\delta t_+ = \pi/8$ and $\delta t_- = \pi/13$ which the agent was trained on are indicated by the vertical lines.

close to unity. There are several reasons for this robust performance: First, in the cases where the agent predicts a wrong action due to sampling noise, it can easily correct for the mistake in subsequent time steps since it learned to prepare the target state from *any* quantum state. Second, although the quantum circuit output is noisy, we find that the subsequent neural network does not enhance the noise and still outputs reasonable values. Finally, since the optimal action is always determined by the argmax of the Q-values, noise in the output does not affect the chosen actions as long as its magnitude is sufficiently small. Hence, we can achieve high success probabilities even in the presence of sampling noise. On the other hand, the average protocol length shown in Fig. 5.9(b) converges to its value under exact computations only after about 10^4 shots which, however, is still in the feasible regime for many modern quantum devices. Note that the performance of the truncated QMPS circuit (green lines in Fig. 5.9) is considerably worse and indicates that we indeed require a bond dimension of $\chi = 4$ to faithfully represent the QMPS agent.

Next, we investigate the effects of noise in the quantum computation on the QMPS framework. To simulate incoherent errors, we consider a depolarizing noise channel E

and apply it after each action and QMPS gate

$$E(\rho) = (1-\lambda)\rho + \lambda\frac{I}{2^N}. \tag{5.16}$$

ρ denotes the quantum state density matrix and λ is the depolarizing noise parameter which is set to $\lambda_1 = 10^{-4}$ for all single-qubit gates. We plot the success rate as a function of the two-and three-qubit gate errors $\lambda_{2/3}$ for 1000 randomly sampled initial states in Fig. 5.10(a). For error rates $\lambda_{2/3} < 10^{-3}$ the QMPS agent is able to successfully prepare the target state in almost all runs. However, the performance quickly deteriorates with increasing error parameter λ. Let us note that we have used the same error rate for both, two-and three-qubit gates. On a physical quantum device, the three-qubit gate will be decomposed into a sequence of two-qubit gates and hence the introduced noise will be amplified. However, the decomposition, the resultant circuit depth, and the sources of noise vary with the hardware type[*]. Thus, we chose the simplified noise model of Eq. (5.16) in order to study the QMPS circuits in a hardware agnostic way.

Finally, we analyze the robustness of the QMPS agent to coherent gate errors similarly to the discussion in Section 5.5.3. Coherent errors arise when the actual, executed gate is different to the gate that has to be applied. These errors can often be mitigated by calibrating the devices carefully. However, frequent calibration is expensive and therefore coherent errors can usually not be eliminated fully. We simulate coherent gate errors for each of the 12 different actions by adding mean-zero Gaussian random noise of standard deviation σ to the time step duration $\delta t_\pm$. In contrast to the discussion in Section 5.5.3, each action gate is fixed, although the angles of rotation δt are shifted compared to the original step size the agent was trained on. We again sample 1000 random initial states and show the state preparation success rates for varying standard deviations σ in Fig. 5.10(b). For each of the 1000 runs we also use a different random seed when sampling the gate noise. We observe that for standard deviations $\sigma < 0.5$, the QMPS agent is still able to self-correct the protocols and reach the target state nonetheless. However, for larger amounts of noise the agent is not capable of reliably preparing the target state anymore. We expect that the performance of the QMPS agent can likely be improved by performing some additional optimization on the quantum device taking into account the exact noise model and error rates.

5.7 Additional details concerning the control studies

Parameters related to the RL environment and the spin systems of each control study can be found in Table 5.1.

We provide some ancillary videos to illustrate the optimal policies learned by the QMPS agent [390]. The description of each video is given below.

Video 1 – Transverse-field Ising control
Bloch sphere trajectory of the reduced density matrix of a single spin in the bulk ($i = 15$) when starting from the initial ground state at $g_x = 1.01$ and acting according

[*]For example, transpiling (i.e., decomposing) the QMPS circuit on an IBM Quantum device results in ~ 100 two-qubit gates, while on an IONQ device we require only ~ 30 two-qubit gates.

Parameter	**Study A** (QMPS-1,QMPS-2)	**Study B**	**Study C**
system size N	4	32	16
single-particle fid. threshold F^*	(0.96, 0.992)	0.99	0.97
max. episode length T	50	50	50
number of actions	12	7	12
step size δt_+	$\left(\frac{\pi}{8}, \frac{\pi}{16}\right)$	$\frac{\pi}{12}$	$\frac{\pi}{12}$
step size δt_-	$\left(\frac{\pi}{13}, \frac{\pi}{21}\right)$	$\frac{\pi}{17}$	$\frac{\pi}{17}$
quantum state bond dim. χ_ψ	4	16	16

Table 5.1: RL environment parameters.

to the optimal QMPS protocol. Also shown are the single-particle and many-body fidelities, the half-chain von Neumann entanglement entropy, the local magnetizations $\langle\sigma_i\rangle$, and the local spin-spin correlations $\langle\sigma_i\sigma_{i+1}\rangle$ during each step of the protocol. The protocol can be divided into three segments involving an initial single-particle rotation and a final many-body Euler-angle-like rotation which reduces unwanted correlations.

Video 2 – Self-correcting Ising control
Bloch sphere trajectories of the reduced density matrix of a single spin at the edge ($i = 0$) and in the bulk ($i = 15$) when starting from the initial ground state at $J = -1, g_x = 1.2, g_z = 0.2$ and preparing the critical target state. The red arrow/curve denotes the trajectory obtained via acting with the optimal QMPS protocol. The blue arrow/curve shows a suboptimal trajectory where at time step $t = 5$ a different than the optimal action is chosen. Afterwards, the systems is again evolved according to the optimally acting QMPS agent. We show two examples of perturbed trajectories consecutively. In both cases the agent is able to successfully prepare the target state despite the perturbation, thus illustrating the ability of the agent to generalize and adapt its protocols on-the-fly.

Video 3 – Self-correcting Ising control
Same as Video 2. However, in this case the blue arrow/curve displays the Bloch sphere trajectory subject to noisy dynamics. Specifically, at each time step we add white Gaussian noise with std $\sigma = 0.05$ to the time step duration $\delta t_\pm$. We show two examples of noisy trajectories with different random seeds consecutively. In both cases the agent is again able to successfully prepare the target state which illustrates the robustness of the QMPS agent to noisy dynamics.

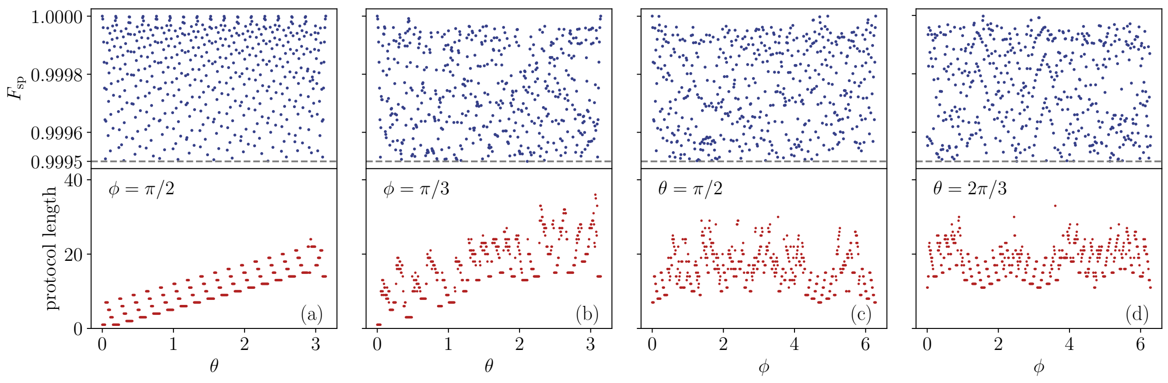


Figure 5.11: Universal single-particle control. Upper panel: Single-particle fidelity between the final and z-polarized target state ($\theta=0$) as a function of the initial state parameters θ and ϕ. In **(a),(b)** ϕ is held fixed; in **(c),(d)** θ is fixed instead. The agent is able to surpass the fidelity threshold $F^*_{\rm sp}=0.9995$, (vertical dashed line) for any state on the Bloch sphere. Lower panel: The corresponding number of protocol steps used by the QMPS agent to reach the target state. $N=64$ spins.

5.7.1 Universal state preparation from arbitrary initial quantum states

Single-particle control

To provide another benchmark of the QMPS framework, we test it in a single-particle control setting. Our goal is to prepare a specific state (here chosen to be the spin-up state) from *any* other single-particle state. We translate this setup to the many-body regime by considering $N=64$ spins uniformly polarized in one direction on the Bloch sphere, which can be exactly approximated with an MPS of bond dimension $\chi_\psi=1$. Note that an arbitrary single-particle spin state can always be expressed as $|\psi\rangle=\cos(\theta/2)|0\rangle+e^{i\phi}\sin(\theta/2)|1\rangle$, where $0\leq\theta\leq\pi$ and $0\leq\phi<2\pi$.

We train a QMPS agent starting each episode from a uniformly sampled state on the Bloch sphere, with a fixed single-particle fidelity threshold of $F^*_{\rm sp}=0.9995$, and an action set that is composed only of single-particle rotations $\mathcal{A}=\{\hat{X},-\hat{X},\hat{Y},-\hat{Y},\hat{Z},-\hat{Z}\}$. Figure 5.11 shows the achieved fidelities between the final and target state for different initial states represented by the angles θ and ϕ. We find in all cases that the QMPS agent is able to successfully reach the fidelity threshold and hence is capable of performing universal single-particle state preparation. We also plot the protocol length starting from each initial state, which, as expected, increases as the distance between the initial and the z-polarized target state ($\theta=0$) becomes larger (see Fig. 5.11(a)).

Universal ground state preparation for $N=4$ spins

In Fig. 5.12 we provide the learning curves of the first QMPS-1 agent which was trained to prepare the Ising ground state with a many-body fidelity threshold of $F^*\sim0.85$ (see Section 5.5.1).

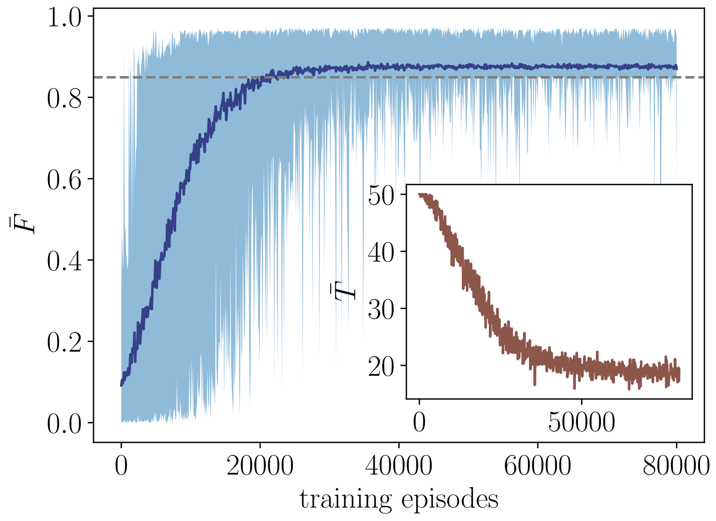


Figure 5.12: Universal four-qubit control. Learning curves of the QMPS-1 agent trained on a many-body fidelity threshold of $F^* \sim 0.85$ (gray-dashed line). The achieved final fidelity $\bar{F}$ is shown averaged over a window of 100 training episodes. The light-blue data indicates the range between the best and the worst fidelity values. Inset: The number of episode steps $\bar{T}$ averaged over 100 episodes. The maximum number of steps per episode was set to 50. $N=4$ spins.

5.7.2 Preparation of a polarized product state from paramagnetic ground states

This section provides further details on the case study presented in Section 5.5.2.

To speed training up, we restrict the action space to 7 actions for this control setup with $\mathcal{A} = \{\hat{Y}, \hat{Z}, -\hat{Z}, \hat{X}\hat{X}, -\hat{X}\hat{X}, \hat{Y}\hat{Y}, -\hat{Y}\hat{Y}\}$. This action set was determined by first training on a smaller system size ($N = 8$) using all actions and then selecting only those that appear in the final optimal protocols.

To get an intuition about the difficulty of this control setup, we proceed as follows: (i) we demonstrate that the initial states are sufficiently far (in the Hilbert space distance) from the target state, e.g., by computing the fidelity as a function of g_x, see Fig. 5.13 (dashed lines). This corresponds to a protocol where the agent does not take any action. (ii) an alternative protocol can be produced by noticing that the initial states are paramagnetic, while the target is a z-polarized state, and thus a $\pi/2$-rotation about the y-axis presents a good candidate for the optimal protocol (it is indeed optimal in the limit $g_x \to \infty$). The corresponding fidelities are shown in Fig. 5.13 (solid lines). Compared to these fidelities, the threshold for the RL agent is given by the horizontal dashed line; it gives a lower bound on the performance of the QMPS protocols. Notice that, the QMPS agents are able to considerably improve on the initial fidelity and outperform the trivial $\hat{Y}$-rotations for the considered range of transverse field values.

The QMPS agent was trained to prepare a specific target state (the z-polarized state) starting from a class of Ising ground states. In principle, the obtained protocols

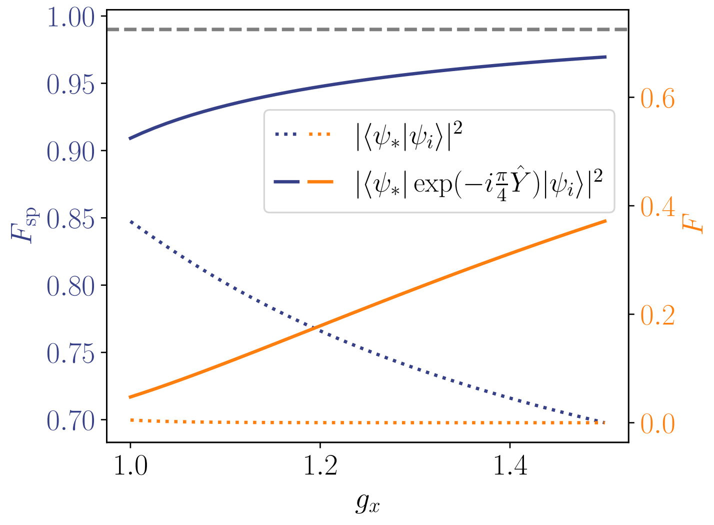


Figure 5.13: Transverse-field Ising control. Single-particle fidelity (blue, left y-axis) and many-body fidelity (orange, right y-axis) between the target z-polarized state and an initial Ising ground state for different values of the transverse field g_x (dotted curves). The solid lines show the respective fidelities of the target state and the initial ground state after applying a single-particle $\hat{Y}$-rotation with $\delta t_- = \pi/4$. The QMPS agent is able to improve on the trivial rotation and prepare the target state with single-particle fidelities $F_{\mathrm{sp}} > 0.99$ (gray dashed line). $N = 32$ spins.

can be inverted to achieve the opposite, i.e. prepare *any* paramagnetic Ising ground state from the z-polarized state. This is often the objective in quantum computing or simulation tasks where the system starts out in a simple product state and is then brought into the state that has to be investigated or that encodes the solution to a problem. With a single trained QMPS agent one can generate optimal controls that, when reversed, prepare a variety of different states that can then be used for computation. Note however, that the final state reached by using the original QMPS protocol does not exactly coincide with the target state since we do not achieve a perfect fidelity of unity. It is therefore not clear whether the inverse protocol, when starting from the exact target state, prepares the original initial states with an equally high fidelity or whether it does considerably worse. In Fig. 5.14 we provide the achieved single-particle fidelities when preparing Ising ground states from the z-polarized state by reversing the optimal QMPS protocols and compare them to the fidelities of the original state preparation routine (see Fig. 5.5 of Section 5.5.2). We find that the fidelities do not differ significantly which justifies that inverse state preparation using the QMPS protocols is possible in this particular control scenario.

In Fig. 5.15 we display three optimal QMPS protocols starting from initial ground states at $g_x = 1.01, 1.1, 1.3$, respectively. Interestingly, in all three cases, the agent learns to initially apply a $\pi/2$-rotation about the y-axis which is decomposed into three consecutive protocol steps since we fix the duration of each applied unitary to be $\delta t_+ = \pi/12$. As discussed in Section 5.5.2, the agent is able to successfully prepare the target state also for initial ground states outside of the training interval, i.e., for

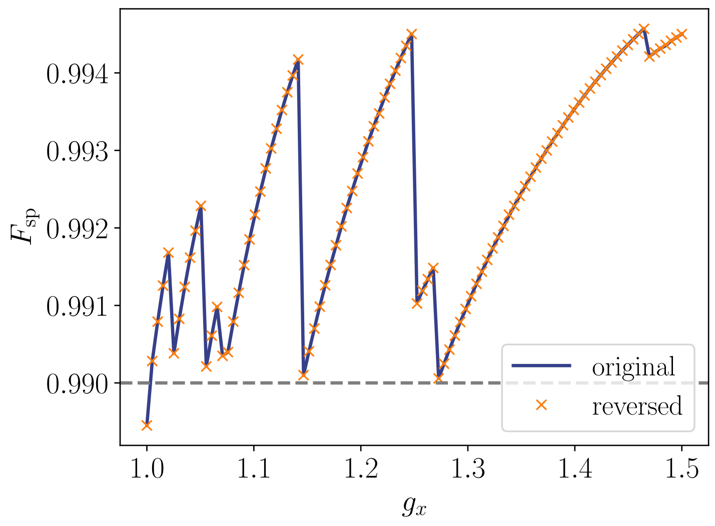


Figure 5.14: Transverse-field Ising control. Final single-particle fidelities when starting from Ising ground states with transverse field values g_x and preparing the z-polarized target state (blue curve). The orange points show the fidelity when reversing the state preparation scenario, i.e. one starts from the polarized state and applies the inverse QMPS protocol to reach the corresponding Ising ground state. The fidelities achieved by the reversed protocol are comparably high. Therefore they justify that, in this case study, the trained QMPS agent can be employed for the inverse state preparation task as discussed in the Section 5.5.2. $N = 32$ spins.

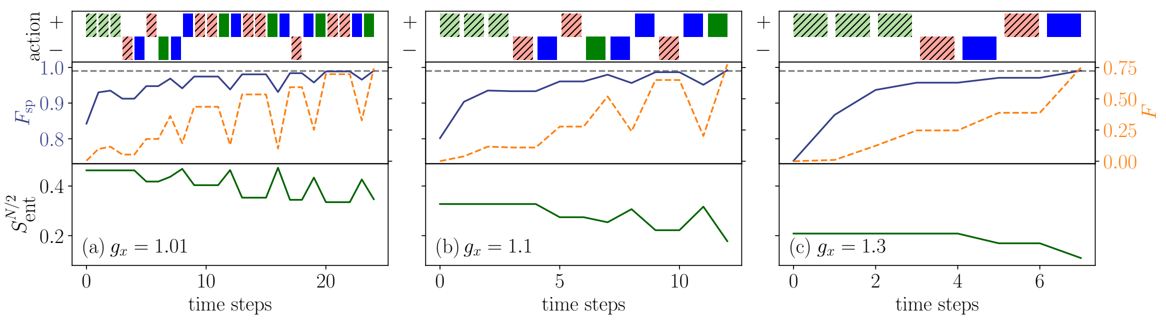


Figure 5.15: Transverse-field Ising control. Upper panels: Final protocols of the QMPS agent when starting from a ground state with **(a)** $g_x = 1.01$, **(b)** $g_x = 1.1$, and **(c)** $g_x = 1.3$. Middle panel: Single-particle fidelities (blue, left y-axis) and many-body fidelities (blue, right y-axis) between the evolved states and the z-polarized target state at each protocol step. The fidelity threshold $F^*_{sp} = 0.99$ is indicated by a gray dashed line. Lower panel: The corresponding half-chain von Neumann entanglement entropy calculated at each step of the protocol. The applied unitaries do not create an excessive amount of entanglement allowing the time evolved states to be described with a relatively low bond dimension. $N = 32$ spins.

$g_x > 1.1$. Note however, that the predicted Q-values of the QMPS agent can be quite different from the true return when tested outside of the training interval, yet the

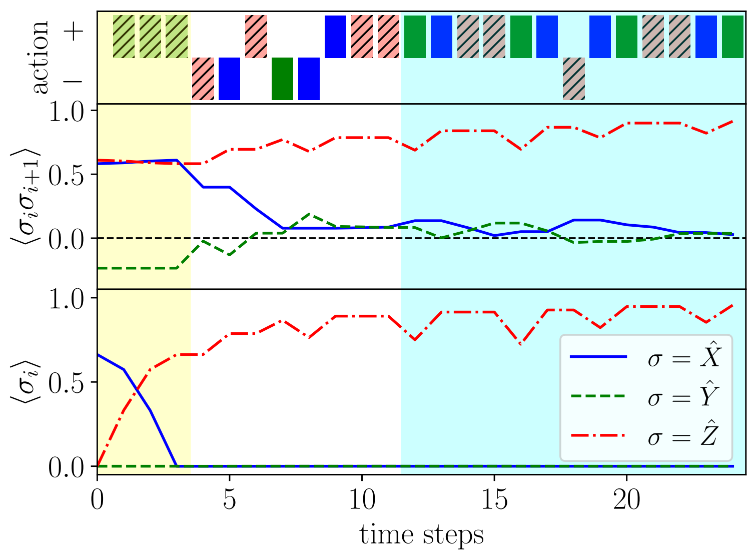


Figure 5.16: Transverse-field Ising control. Analysis of the optimal QMPS protocol obtained when starting from the initial ground state at $g_x = 1.01$. Shown are the local spin-spin correlations $\langle\sigma_i\sigma_{i+1}\rangle$ and the local magnetization $\langle\sigma_i\rangle$ along each direction at the center of the spin chain ($i = 15$). The yellow shaded segment indicates the initial $\hat{Y}$-rotations which align the spin along the z-axis; the blue shaded area points to a generalized Euler-angle-like many-body rotation which reduces unwanted correlations and disentangles the state. $N = 32$ spins.

policy learned by the agent can still produce meaningful optimal protocols. In the bottom panels of Fig. 5.15 we show the half-chain von Neumann entanglement entropy S_{ent} of the encountered states when evolving according to the optimal protocols. The entanglement entropy stays small and hence, allows the time evolved system to be simulated with a relatively small bond dimension χ_ψ (for training we set $\chi_\psi = 16$). Note however, that the entanglement entropy is not fully reduced to zero at the end of the protocol which is especially the case for the states close to the critical point. This can be attributed to the logarithmic scaling of the entanglement entropy in the initial critical state, with the subsystem size $N/2$ due to the presence of long-range correlations. While the agent is able to reduce local correlations effectively (see Fig. 5.16, middle panel), the short protocol is not capable of destroying all long-range correlations which persist in the final state. The prepared state, therefore, has a finite many-body overlap with the separable target state (see orange curves/axis in Fig. 5.15 showing the many-body fidelity), which explains the discrepancy in the final entanglement entropies.

In Fig. 5.16 we plot the local expectation value of the magnetization along each direction and the local spin-spin correlations at the center of the chain which reveal the role of each unitary occurring in the protocol sequence corresponding to the state preparation task shown in Fig. 5.15(a) ($g_x = 1.01$). As already mentioned, the first three $\hat{Y}$-rotations align the state along the z-axis bringing the expectation values of the $\hat{X}$ and $\hat{Y}$ component to zero. The role of the remaining unitaries is to decrease unwanted correlations and consequently to disentangle the state. Note that the $\langle\hat{X}\hat{X}\rangle$ and $\langle\hat{Y}\hat{Y}\rangle$ correlations approach zero in an intertwined manner which is caused by an intricate

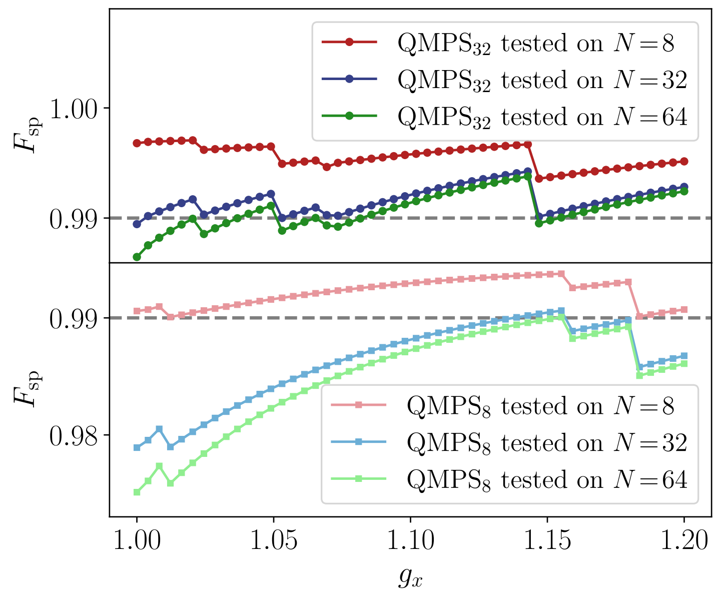


Figure 5.17: Transverse-field Ising control. Upper panel: Final single-particle fidelities achieved when testing the protocols of a QMPS agent trained on $N=32$ spins (blue curve) on a smaller system size of $N=8$ (red curve) and on the larger $N=64$ spin system (green curve). The increase in the single-particle fidelity for $N=8$ suggests that optimized protocols can be transferred to smaller system sizes for this particular control setup. Lower panel: The opposite scenario, where the protocols of a QMPS agent trained on $N=8$ spins (light red) are tested on larger systems of $N=32$ spins (light blue) and $N=64$ spins (light green). The fidelity threshold (gray dashed line) cannot always be maintained for the larger system size especially close to the critical point at $g_x \sim 1$.

combination of $\hat{X}\hat{X}$ and $\hat{Y}\hat{Y}$ disentangling gates and single-particle $\hat{Z}$-rotations. The latter are important for finely realigning the state after each disentangling operation and as such preventing the correlations from diverging.

Next, we analyze how well the optimal protocols perform on systems with a different number of spins. To this end, we test the protocols optimized for $N=32$ spins (QMPS$_{32}$) on $N=8,64$ spin systems, and vice-versa: protocols obtained after training on $N=8$ spins (QMPS$_8$) are tested on the larger $N=32,64$ systems (see Fig. 5.17). We find that the fidelity threshold can still be reached when applying the QMPS$_{32}$ protocols on smaller system sizes. However, the opposite is not true: the QMPS$_8$ protocols, in general, give rise to fidelities below the threshold when tested on the $N=32,64$ spin systems. These system-size (in)dependence suggests that, for this particular control setup, one can devise suitable pretraining techniques for large system sizes, based on the behavior of agents successfully trained on smaller systems. Moreover, the QMPS agent tends to find optimal protocols which appear robust to changes in the system size, and the control task likely admits a solution also in the thermodynamic limit.

Finally let us comment on the quantum state entanglement and the related MPS bond dimension. During the initial exploratory stage of training, random unitaries

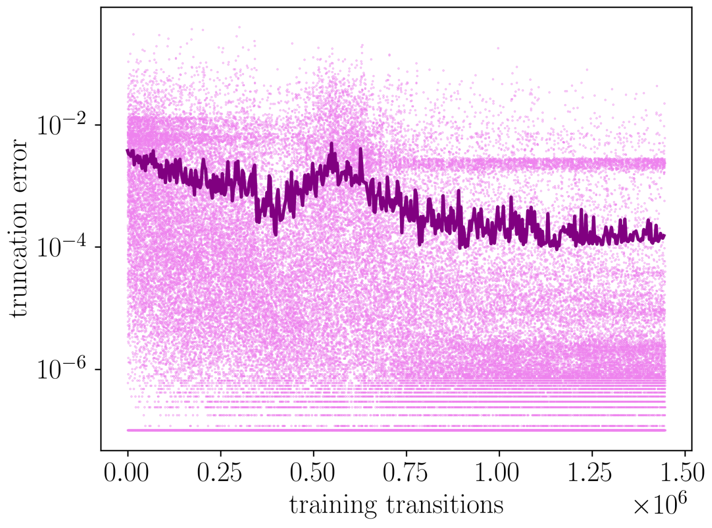


Figure 5.18: Transverse-field Ising control. Truncation error for the evolved states encountered during training of the QMPS agent. The dark purple curve shows the truncation error averaged over a window of 3000 transitions. Training was performed with a quantum state bond dimension of $\chi_\psi = 16$; $N = 32$ spins.

are applied to the system leading in general to a ballistic growth of the entanglement entropy. In this case the fixed bond dimension of $\chi_\psi = 16$ is not always sufficient to capture the evolved quantum states giving rise to the large truncation errors, shown in Fig. 5.18. These truncation errors, however, naturally decrease as training progresses and the action selection becomes more deterministic while the Q-function converges close to the optimal one. While for training a bond dimension of $\chi_\psi = 16$ was used, testing was performed with $\chi_\psi = 32$ for which the truncation errors vanished to machine precision. This check ensures the stability of the QMPS protocols to changes in the accuracy of the MPS approximation.

5.7.3 Learning robust critical-region state preparation

This section provides further details on the case study presented in Section 5.5.3.

To compare the achieved fidelities of the QMPS agent reported in Fig. 5.6(a), we show the fidelities between the initial mixed-field Ising ground states and the critical target state before any controls are applied in Fig. 5.19. The QMPS agent is able to reach the target with single-particles fidelities $F_{\text{sp}} > 0.97$ (corresponding to a many-body fidelity of $F \sim 0.61$).

The half-chain von Neumann entanglement entropy of the quantum states during training is displayed in Fig. 5.20. Similar to the previous case study, during the initial stages of training the encountered states are highly entangled due to the randomness of the action selection. Once the agent learns how to reliably prepare the target state, the entropies decrease and are scattered closely around the target state value (orange dashed line). We emphasize that critical states possess a logarithmic correction to the area-law of entanglement, which makes their preparation a non-trivial task. For

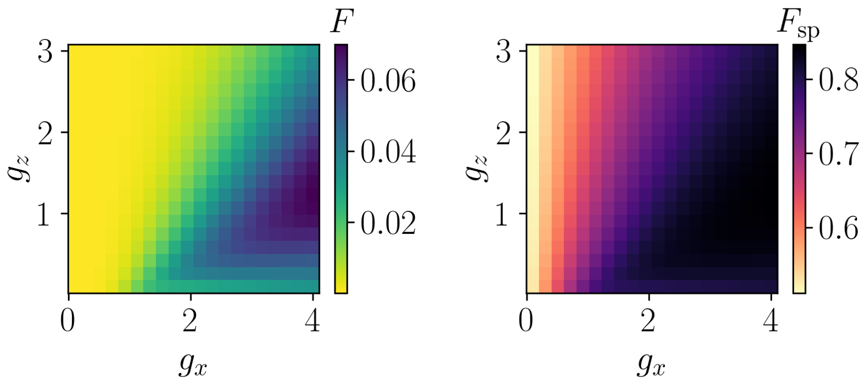


Figure 5.19: Self-correcting mixed-field Ising control. Many-body fidelity F (left) and single-particle fidelity F_{sp} (right) between the critical target state and the initial ground states at transverse and longitudinal field values g_x, g_z. $N = 16$ spins. The trained QMPS agent prepares the target state with many-body and single-particle fidelities $F > 0.61, F_{\text{sp}} > 0.97$ respectively, and therefore considerably improves on the initial fidelity values. $N = 16$ spins.

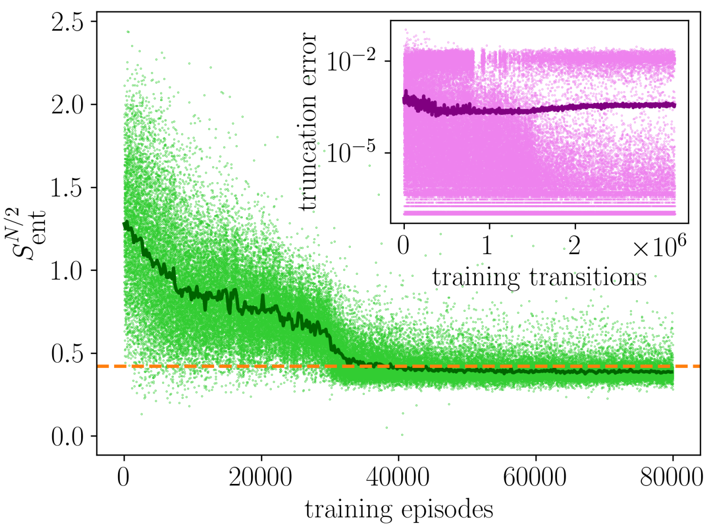


Figure 5.20: Self-correcting mixed-field Ising control. Half-chain von Neumann entanglement entropy of final states during training. The dark green curve denotes the average over 200 episodes and the orange dashed line indicates the entanglement entropy of the critical target state. The entropies decrease as learning progresses and converge to a value close to that of the target state. Inset: Truncation error averaged over a window of 3000 transitions (dark purple). Training was performed with a quantum state bond dimension of $\chi_\psi = 16$; for testing $\chi_\psi = 32$ was used. $N = 16$ spins.

training we used a relatively small bond dimension of $\chi_\psi = 16$ which led to the finite truncation errors shown in the inset of Fig. 5.20. However, training is still successful and all subsequent tests were performed by setting $\chi_\psi = 32$ which did not affect the

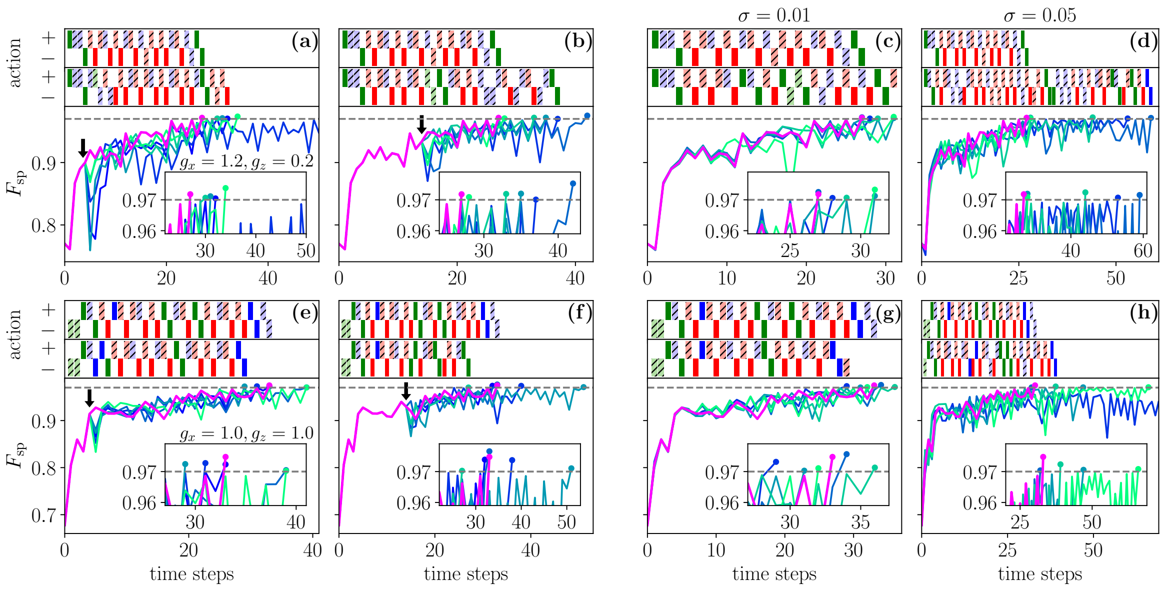


Figure 5.21: Self-correcting mixed-field Ising control. Exemplary QMPS protocols and time-dependence of the single-particle fidelity when starting from an initial ground state within the training region at $J=-1, g_x=1.2, g_z=0.2$ **(a)-(d)**, and outside the training region at $J=-1, g_x=1.0, g_z=1.0$ **(e)-(h)**. For each subplot the upper panel displays the optimal QMPS protocol without perturbations, the middle panel presents an exemplary protocol subject to noise or perturbations, and the bottom panel shows the single-particle fidelities at each time step for different protocols (the original, unperturbed QMPS protocol is always indicated by the magenta line). The single-particle fidelity threshold of $F^*_{\text{sp}}=0.97$ is denoted by a gray dashed line. In **(a)-(b), (e)-(f)** the QMPS protocol is modified at time step $t=5$ ($t=15$) [indicated by a black arrow] by taking 5 different random actions. Afterwards, the system is again evolved according to the QMPS agent leading to 5 distinct trajectories (blue lines). In all but one case the QMPS agent is able to correct for the mistake and successfully reaches the fidelity threshold. In **(c)-(d), (g)-(h)** white Gaussian noise with standard deviation $\sigma=0.01$ **(c),(g)**, or $\sigma=0.05$ **(d),(h)** is added to the time step duration $\delta t_\pm$. The system is evolved with 5 different random seeds (blue lines). The QMPS agent is again able to adapt its protocol and successfully reaches the fidelity threshold in most cases. $N=16$ spins.

QMPS protocols or the final achieved fidelities.

Next, we provide further examples showcasing the ability of the QMPS agent to self-correct its protocols on-the-fly when the time evolution is noisy or perturbed. In Fig. 5.21, we consider two different initial ground states, one within the training region ($J=-1, g_x=1.2, g_z=0.2$) [(a)-(d)] and one outside ($J=-1, g_x=1.0, g_z=1.0$) [(e)-(f)], and analyze the success of state preparation subject to different protocols. In the upper panels of each subplot in Fig. 5.21 we display the actions of the optimal protocol (top) and one exemplary protocol that has been perturbed (bottom). The lower panel always shows the single-particle fidelities at each step of the protocols.

First, we take the optimal QMPS protocols and modify it at time step $t=5$ ($t=15$)

by taking 5 random suboptimal actions instead. Afterwards the system is again evolved according to the greedily acting QMPS agent giving rise to 6 distinct trajectories (the magenta line corresponds to the unperturbed one). In most cases the agent is able to correct for the mistake by adapting the subsequent protocol and reaches the fidelity threshold nonetheless. However, in one instance (see Fig. 5.21(a)), the resultant QMPS protocol does not converge and the agent fails to prepare the target state. Hence, the agent is not able to generalize to states generated by this particular protocol sequence, and likely predicts wrong Q-values that steer the agent eventually away from the target state. This is, however, not surprising since the agent has only been trained on states within a small part of the many-body Hilbert space and therefore, it cannot be expected to devise successful protocols from arbitrary states.

Note that for the initial state outside of the training interval [(e),(f)], the perturbation of the original QMPS protocol gave rise to a shorter protocol, i.e. the fidelity threshold is reached in a fewer number of steps. Hence, in this case the original protocol is not a local minimum of the control landscape. However, this is not surprising, since the QMPS agent has not been trained on this initial state and therefore, the predicted Q-values are not guaranteed to have converged to the true optimal values.

Finally, we study the robustness of the QMPS agent to a randomized time step duration $\delta_\pm$ by adding white Gaussian noise with standard deviation $\sigma = 0.01, 0.05$ to it (see Fig. 5.21(c),(d),(g),(h)). We evolve the system with 5 different random seeds giving again rise to 6 distinct trajectories (the magenta line corresponds to the unperturbed one). For each of the 5 randomized time evolutions, the QMPS agent has to eventually adapt its protocol by performing a different sequence of actions. It successfully prepares the target state in all but one case which falls outside the training region (see Fig. 5.21(h)). Note here as well that in some instances the agent is able to devise shorter control protocols compared to the original unperturbed ones. Hence, the randomized step duration can have a positive effect on the control problem allowing for faster state preparation protocols.

In Fig. 5.22 we compare the achieved fidelities in the presence of noise when we evolve with the adapted protocols (bottom) and with the original, noise-free protocol (top) starting from an initial state $J=-1, g_x=1.2, g_z=0.2$ within the training region. We again consider Gaussian random noise with standard deviations $\sigma = 0.01, 0.05$ and repeat the time evolution with 100 different random seeds for each of the two cases. When the noise is weak ($\sigma = 0.01$), the fixed, unperturbed protocol gives rise to qualitatively similar fidelity curves regardless of the seed (see Fig. 5.22(a)). We found that for 90 out of the 100 runs, the protocol successfully prepares the target state, that is, the fidelity threshold is reached within 50 steps. However, in the cases where the fidelity threshold is not surpassed, the final fidelities are close to the target value of $F^*_{\mathrm{sp}} = 0.97$. If we instead allow the QMPS agent to adapt to the perturbed states, we obtain the fidelity curves in Fig. 5.22(b). The resultant protocols give rise to qualitatively different trajectories that diverge more towards later time steps (compare to Fig. 5.21(a)-(d)). However, in this case the agent successfully prepares the target state for each of the 100 runs. Hence, for sufficiently weak noise strengths, the original unperturbed protocol is expected to give qualitatively similar results to the noise-free dynamics. However, even in this example the self-correcting agent has a measurable advantage over the fixed protocol.

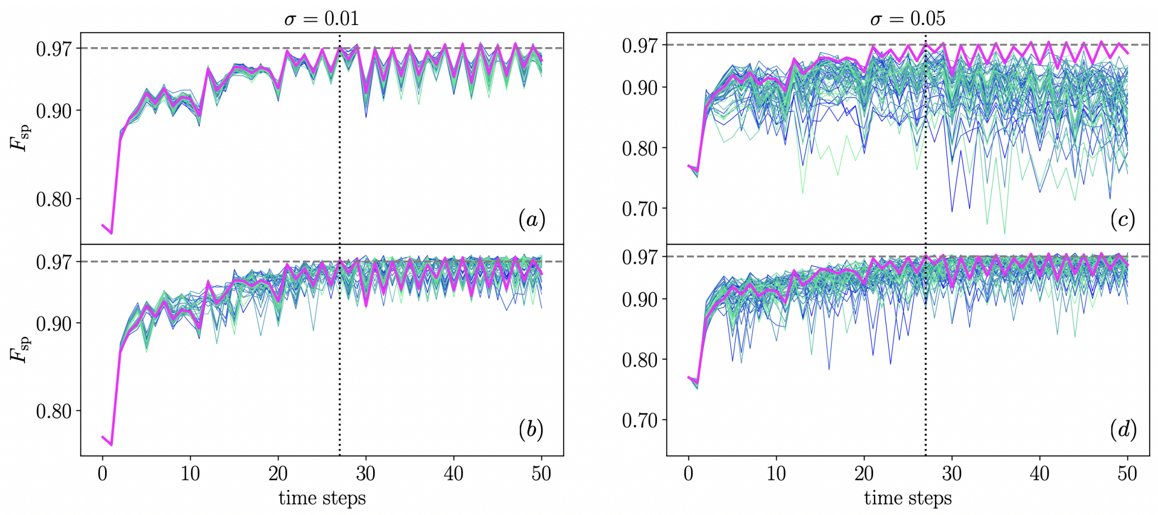


Figure 5.22: Self-correcting mixed-field Ising. Time-dependence of the single-particle fidelity when adding Gaussian random noise with standard deviations $\sigma = 0.01$ **(a)-(b)**, and $\sigma = 0.05$ **(c)-(d)** to the time step duration $\delta t_{\pm}$ and starting from an initial ground state at $J = -1, g_x = 1.2, g_z = 0.2$. For each figure, we sampled 100 different trajectories (each corresponding to a different random seed). The original, unperturbed QMPS protocol is always indicated by the magenta line. The single-particle fidelity threshold of $F^*_{\text{sp}} = 0.97$ is denoted by a horizontal gray dashed line and the number of steps required to reach the threshold in the noise-free case is indicated by a vertical black dotted line. In **(a)**,**(c)** we always evolve according to the fixed, unperturbed QMPS protocol we obtained from the noise-free simulation. The percentage of successfully prepared target states within 50 steps is 90% in the case of weak noise ($\sigma = 0.01$) and 0% in the case of strong noise ($\sigma = 0.05$). In **(b)**,**(d)** we use the adaptive QMPS agent to generate different protocols for each distinct run (compare to Fig. 5.21(a)-(d)). The respective success percentages are 100% ($\sigma = 0.01$) and 74% ($\sigma = 0.05$). Hence, in both instances, the self-correcting agent is able to improve over the fixed, noise-free protocol.

This situation changes when we consider the case of strong noise ($\sigma = 0.05$) as shown in Fig. 5.22(c)-(d). The fixed, unperturbed protocol leads to diverging fidelity curves already after a few steps (top). In fact, the success probability for the simulated 100 runs is 0. In contrast, the adaptive agent prepares the target state successfully for 74 out of the 100 instances within the 50 allowed time steps. Moreover, the agent clearly tries to steer the quantum states towards high fidelity regions. This example therefore demonstrates that the self-correcting agent is able to improve over the original, noise-free protocol when the dynamics is being perturbed.

5.8 Details of the QMPS architecture and training

In all examples discussed in this chapter the QMPS tensors are initialized as identity matrices with Gaussian noise ($\sigma = 0.2$) added to all components both for the real and complex parts. The tensors are additionally scaled by a factor of 0.25. The neural

Parameter	**Value**
number of training episodes N_{eps}	40000 – 80000
optimizer	ADAM
learning rate α	$5 \times 10^{-5} - 1 \times 10^{-4}$
batch size	32 – 64
RL discount factor γ	0.98
RL buffer size	8000
target network update frequency n_{target}	10
initial exploration ϵ_{init}	1.0
final exploration ϵ_{final}	0.01
exploration decay ϵ_l	$\exp(-8 \times l/N_{\text{eps}})$
QMPS bond dimension χ_Q	4 – 32
QMPS feature vector dimension d_f	32 – 72
NN number of hidden layers	2
NN number of hidden neurons	100 – 200
NN nonlinearity	tanh

Table 5.2: QMPS training hyperparameters.

network weights and biases are initialized with real Gaussian random numbers ($\sigma = 0.1$). All parameter values of the QMPS framework are summarized in Table 5.2. The values of the hyper-parameters including the time evolution step sizes $\delta t_\pm$ are obtained by performing a coarse grid-search, i.e. we trained on a few different parameter values and select the ones which yield best performance results. Note that we adopt slightly different values for δt_+ and δt_-. This choice prevents the agent to simply undo an action by evolving with the inverse operator which helped stabilise training.

As mentioned in Secion 5.5.1 a single QMPS agent is not able to reach arbitrarily high fidelities due to the discreetness of the action space, the constant step size, and the fixed maximum episode length. An additional challenge is posed by the large deviation in the expected return values for states at the beginning and the end of the episode: The QMPS network is not able to resolve small differences in the reward which is however required close to the target state where the log fidelities approach zero. Therefore, we introduce a multi-stage learning scheme in Section 5.5.1 where successive agents with tighter fidelity thresholds are trained starting from states which are pre-prepared from agents optimized on smaller thresholds. This training strategy also allows the step size to be chosen separately for each agent.

We show the pseudocode of the DQN algorithm used for training the QMPS agent in Algorithm 1. The corresponding Python code can be found on GitHub [391].

Algorithm 1 QMPS training

```
Input: Target state $|\psi_*\rangle$, fidelity threshold $F^*$, maximum episode length $T$, number of
    training episodes $N_{\text{eps}}$, learning rate $\alpha$, batch size $N_{\text{batch}}$, discount factor $\gamma$, replay
    buffer size $N_{\text{buff}}$, target network update frequency $n_{\text{target}}$, exploration parameters
    $(\epsilon_{\text{init}}, \epsilon_{\text{final}})$
 1: Initialize QMPS network $Q_\theta$ and copy parameters to target network $\bar{\theta} \leftarrow \theta$
 2: Reset RL environment (sample initial state $|\psi_0\rangle$)
 3: # Fill replay buffer with random transitions
 4: for $i = 1, .., N_{\text{buff}}$ do
 5:     Select random action $a \rightarrow \pm\hat{A}$
 6:     Time evolve state $|\psi'\rangle = \exp\Big(\pm i\delta t_\pm \hat{A}\Big)|\psi\rangle$
 7:     Compute reward $r = N^{-1}\log(|\langle\psi'|\psi_*\rangle|^2)$
 8:     Append transition $(|\psi\rangle, a, r, |\psi'\rangle)$ to replay buffer
 9:     Set $|\psi\rangle = |\psi'\rangle$
10:     if $r > F^*$ or $T$ is reached then
11:         Reset RL environment (sample new initial state $|\psi_0\rangle$)
12: # Start training
13: for $l = 1, .., N_{\text{eps}}$ do
14:     Reset RL environment (sample initial state $|\psi_0\rangle$)
15:     Compute decay exploration parameter: $\epsilon_l = \epsilon_{\text{final}} + (\epsilon_{\text{init}} - \epsilon_{\text{final}})\exp(-8l/N_{\text{eps}})$
16:     for $t = 0, .., T$ do
17:         # Update network
18:         Sample $N_{\text{batch}}$ transitions $(|\psi\rangle, a, r, |\psi'\rangle)$ from replay buffer
19:         Compute regression target $y = r + \gamma\ Q_{\bar{\theta}}(\psi', \text{argmax}_{a'}\ Q_\theta(\psi', a'))$
20:         Compute gradients of $L(\theta) = \sum_{\text{batch}}(y - Q_\theta(\psi, a))^2$ w.r.t. parameters $\theta$
21:         Perform gradient descent step using ADAM
22:         Every $n_{\text{target}}$ steps: Copy QMPS parameters $\theta$ to target QMPS network
    $\bar{\theta} \leftarrow \theta$
23:         # RL environment step
24:         Select action $\pm\hat{A}_t \leftarrow a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon_l \\ \text{argmax}_a Q_\theta(\psi_t, a) & \text{otherwise} \end{cases}$
25:         Time evolve state $|\psi_{t+1}\rangle = \exp\Big(\pm i\delta t_\pm \hat{A}_t\Big)|\psi_t\rangle$
26:         Compute reward $r_t = N^{-1}\log(|\langle\psi_{t+1}|\psi_*\rangle|^2)$
27:         Append transition $(|\psi_t\rangle, a_t, r_t, |\psi_{t+1}\rangle)$ to replay buffer
28:         if $r_t > F^*$ or $T$ is reached then BREAK
```

Optimization

The gradients of the neural network and the QMPS parameters can be computed via conventional backpropagation and, in principle, any automatic differentiation library can be employed for this task. However, we obtained a considerable speed-up (factor of ~ 10) by implementing the gradient computation from scratch. The neural network takes as input the real-valued QMPS feature vector and therefore the parameters are chosen to be real-valued as well. On the other hand, restricting the QMPS tensors to real numbers greatly limited the expressivity of the ansatz. Hence, each tensor component is comprised of both a real and imaginary parameter. Due to the overall QMPS ansatz being not holomorphic (the absolute value is not complex differentiable), the real and imaginary parameters have to be updated independently by computing the gradient with respect to each of them separately.

Compute resources

For a system of size N, local Hilbert space dimension d, and uniformly fixed bond dimension χ, the number of MPS parameters scale as $N\chi^2 d$. The quantum state MPS time evolution (based on SVD and matrix multiplication) as well as the QMPS optimization (based only on matrix multiplication) scale linear in N and at worst polynomial in χ and d. We have not fixed the bond dimensions of the MPS and QMPS to be uniform on all sites, but rather let both of them grow exponentially from the boundary up to a maximum uniform bond dimension in the middle of the MPS. For the hyperparameters chosen in this study, most time was spent in the optimization step which requires two forward passes and one backward pass on a batch of input states. Overall, one full episode of training (including 50 environment and optimization steps) for $N = 16, \chi_Q = 32, \chi_\psi = 16, d_f = 32$, and a batch size of 64 took 6.5 sec on a Intel Xeon Gold 6230 CPU and 1.8 sec on a NVIDIA Tesla P100 SXM2 GPU. Let us note that the code has not been optimized for a GPU and with some modifications an even larger speedup can be expected. Therefore, larger system sizes should also be within reach in the near future.

5.9 Conclusion

In this work we introduced a tensor network-based Q-learning framework to control quantum many-body systems. Incorporating an MPS into the deep learning architecture allows part of the Q-value computation to be efficiently expressed as an overlap between two MPS wave functions. As a result, larger system sizes can be reached compared to learning with the full wave function. We emphasize that standard RL algorithms with conventional neural network architectures cannot handle quantum many-body states, whose number of components scale exponentially with the number of spins: e.g., for $N = 32$ spins, there are $2^{32} \approx 10^{10}$ wavefunction components to store which is prohibitively expensive. By contrast, our MPS learning architecture only requires linear scaling with the system size N. Furthermore, we found that the hyperparameters of the optimization and, in particular, the number of training episodes do not require finetuning with the system size, and stayed roughly constant (see Section 5.8). Summarizing, QMPS proposes the use of a tensor-network variational ansatz inspired by quantum many-body physics to offer a novel RL learning architecture.

QMPS-based RL is designed for solving the quantum many-body control problem by learning a value function that explicitly depends on the quantum state. Therefore, a successfully trained QMPS agent is capable of devising optimal protocols for a continuous set of initial states, and selects actions on-the-fly according to the current state visited. As a result, a QMPS agent has the ability to self-correct mistakes in the protocols when the dynamics is stochastic, even before the protocols have come to an end. Moreover, we illustrated that the agent can interpolate and extrapolate to new quantum states not seen during training. Remarkably, we observed this behavior over test regions several times the size of the training region. To the best of our knowledge, there does not exist a quantum control algorithm that exhibits such desirable features, as these are based on deep learning capabilities: conventional quantum control algorithms require to re-run the optimization when the initial state has been changed, and thus lack any learning capabilities.

The generalization capabilities, the robustness to noise, and the feasibility of universal state preparation (for small system sizes) are unique advantages of the QMPS framework over competitive optimal control algorithms. These features are especially relevant for experiments and modern quantum technologies that heavily rely on quantum many-body control, and in particular for NISQ devices. We demonstrated that the present QMPS framework can be integrated in quantum device simulations by mapping the optimized MPS ansatz to gates in a quantum circuit. The resultant hybrid quantum-classical algorithm allows us to control quantum states directly on the device without the need of performing expensive quantum state tomography. Thus, an adaptive on-line agent can be trained on the noisy hardware and mitigate the effects of errors at each step of the computation.

Our work opens up the door to further research on tensor network-based RL algorithms for quantum (many-body) control. Due to the modular structure of the architecture, the QMPS can be replaced by various tensor networks, such as tree tensor networks (TTN) [117] or the multi-scale entanglement renormalization ansatz (MERA) [118]; these would allow different classes of states to be represented efficiently, and affect the expressivity of the ansatz. It is also possible to use a matrix product operator (MPO) in place of the MPS, which would require us to calculate an expectation value rather than a fidelity [129]. Furthermore, if the operators can be made fully local, i.e., by acting on only two or three sites, we can interpret the trainable parameters as representing a local observable which we can efficiently measure on present-day quantum devices. Another potential enhancement would be to build symmetries (e.g. reflection, translational, etc.) into the tensor network ansatz which could ultimately reduce the required computational resources. Furthermore, we can also consider controlling infinite-sized systems and use the iMPS and iDMRG formalism to simulate and parameterize the agent instead [36].

Tensor networks come with a comprehensive toolbox for analyzing their properties, such as the entanglement structure and correlations. Hence, tensor-network-based reinforcement learning will enable us to study the data, the training, and the expressivity of the ansatz using well-understood concepts from quantum many-body physics [38, 392]. Tensor networks and MPS in particular also allow for different optimization strategies that are not based on traditional backpropagation [33]. In analogy to DMRG, we can compute gradients and update tensors locally at only one or two sites at a time. Moreover, matrix decompositions like the SVD allow us to adaptively choose the bond dimension and with that the number of optimizable parameters as part of the training. It is therefore an interesting future direction to compare the different optimization approaches in this RL setting.

In terms of the computational performance, there are several possible improvements that would potentially allow for faster training. Currently, the main bottleneck is the optimization of the QMPS network. While it is crucial to choose a large enough bond dimension when time evolving the quantum states, one can truncate each quantum state MPS before feeding it into the Q-value ansatz. It would be interesting to see to what extent the quantum states can be compressed for training to still be successful. Other possibilities for speeding up training are to use a pre-filled replay buffer, e.g., with optimal transitions obtained for smaller system sizes, or to pretrain the QMPS agent in a supervised manner on said transitions.

Note that it is straightforward to use RL algorithms other than Q-learning in conjunction with our MPS-based ansatz. For instance, policy-gradient methods instead of Q-learning allow for continuous action spaces, and hence target states can be reached with higher fidelity [211]. Similarly, we can relax the constraint on the actions space of applying each operator uniformly to all spins and instead allow the applied gates to be different at each site which is usually the case for quantum computing applications. However, the dimension of such an action space would grow linearly with the number of particles, and therefore, training would become infeasible in the many-body regime when using the current tools. Another interesting direction of this work is to employ multi-task reinforcement learning to learn a goal-dependent policy [393]. This would allow us to train a single agent to prepare a family of target states and hence, solve a whole class of control problems at the same time.

Finally, one can also adapt the reward function and, for instance, consider the energy density, various distance measures beyond the fidelity, or a completely different objective function such as the entanglement entropy. It would also be interesting to apply the QMPS control framework to a wide range of different systems, such as, Heisenberg models, Hubbard models, time-dependent Hamiltonians, and systems in higher dimensions (e.g. using the 2d PEPS analog of an MPS [130]).

Conclusion

In this thesis, I have discussed four different applications of machine learning techniques to problems in quantum physics. I have showcased how the three machine learning types – supervised, unsupervised, and reinforcement learning – can each be leveraged for diverse tasks ranging from vortex detection in rotating Bose-Einstein condensates, to anomaly detection for unsupervised phase discovery, and to quantum control. To that end, I have employed both, conventional neural network models, and tensor network-based approaches which are tools inspired by computational quantum many-body physics. Furthermore, I have demonstrated how present-day quantum computers can be effectively harnessed for machine learning tasks on quantum data.

In the following I will briefly summarize each of the four main projects of this thesis, which resulted in three peer-reviewed publications and one preprint article which is currently under review. For a detailed discussion of each study I refer to the corresponding chapter in the thesis.

Chapter 2: Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates

I developed a vortex detection framework based on convolutional neural networks that can accurately locate all vortices in density images of atomic Bose-Einstein condensates (BECs). The technique works both on groundstate and on more challenging non-equilibrium configurations. Furthermore, the vortex detector gives accurate predictions even in the presence of different sources of noise. Hence, it is especially suited for post-processing noisy images obtained from experiments. Additionally, the model trained on snapshots of BECs in harmonic traps can generalize to images where the BEC is trapped in a ring-shaped potential instead. I further showed that with minor modification to the model, the vortex detector can also learn to classify the circulation direction of each vortex when the phase profile is provided as well. This chapter has been published in: Machine Learning: Science and Technology **2**, 035019 (2021).

Chapter 3: Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer

I and my collaborators proposed an unsupervised quantum machine learning framework for discovering phases of quantum many-body systems. The variational quantum algorithm is based on the idea of anomaly detection and implemented using a quantum autoencoder architecture. It allows quantum systems that are simulated on a quantum computer to be investigated end-to-end on the same device in an automated fashion without requiring prior knowledge about the system at hand. We demonstrated the framework on the paradigmatic transverse-longitudinal field Ising model and on the extended Bose Hubbard model with dimerized hoppings which hosts a symmetry protected topological phase. In each case our anomaly detection scheme was able to successfully map out the respective phase diagrams using classical simulators. Furthermore, we showed that the algorithm can already be applied on existing noisy quantum hardware and executed it on an IBM Quantum device. This chapter has been published in: Phys. Rev. Research **3**, 043184 (2021).

Chapter 4: Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks

I and my co-authors investigated different approaches for generating highly entangled states independent of the initial state in 1D discrete-time quantum walks. First, we proposed a deterministic sequence of coin operators comprised of the Hadamard and Fourier coin that leads to high amounts of hybrid entanglement irrespective of a class of localized initial states. We showed that the universal entangling coin sequence works for any odd number of time steps and derived its asymptotic limit for an infinitely-long quantum walk. We then compared the Schmidt norms achieved by the deterministic coin sequence to those obtained from an optimization approach. To that end, we used Q-learning to maximize the hybrid entanglement in a quantum walk over all localized initial states. We found that the optimized coin sequences give rise to higher average Schmidt norms than the deterministic coin sequences. However, the former varies with respect to the initial state parameters opposed to the Schmidt norms attained from the deterministic sequence. This chapter has been published in: Journal of Physics A: Mathematical and Theoretical **53**, 445306 (2020).

Chapter 5: Self-correcting quantum many-body control using reinforcement learning with tensor networks

I developed a new reinforcement learning framework for controlling quantum many-body systems. The approach is based on matrix product states which are leveraged (i) for efficiently simulating the quantum many-body wave function and (ii) as an efficient machine learning ansatz for the Q-learning agent (QMPS). This allowed us to reach system sizes that are otherwise inaccessible with exact techniques and neural network-only approaches. To demonstrate the QMPS framework, me and my collaborator considered the mixed field Ising model and three different ground state preparation problems. First, I showed that for small system sizes of $N = 4$ spins, a single QMPS agent can learn to prepare a target state from arbitrary, random initial states and hence perform universal state preparation. Further, I demonstrated that in certain situations an agent can generalize and extrapolate its optimal protocols to states not encountered during training. As a result, the agent is able to successfully devise and adapt its protocols when the quantum dynamics becomes noisy or stochastic. This chapter is based on the preprint article: arXiv:2201.11790 [quant-ph] (2022).

Bibliography

[1] F. Metz, J. Polo, N. Weber, and T. Busch, *Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates*, Machine Learning: Science and Technology **2**, 035019 (2021).

[2] K. Kottmann, F. Metz, J. Fraxanet, and N. Baldelli, *Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer*, Phys. Rev. Research **3**, 043184 (2021).

[3] A. Gratsea, F. Metz, and T. Busch, *Universal and optimal coin sequences for high entanglement generation in 1D discrete time quantum walks*, Journal of Physics A: Mathematical and Theoretical **53**, 445306 (2020).

[4] F. Metz and M. Bukov, *Self-Correcting Quantum Many-Body Control using Reinforcement Learning with Tensor Networks*, arXiv e-prints , arXiv:2201.11790 (2022).

[5] R. Sachdeva, F. Metz, M. Singh, T. Mishra, and T. Busch, *Two-leg-ladder Bose-Hubbard models with staggered fluxes*, Phys. Rev. A **98**, 063612 (2018).

[6] K. Gietka, F. Metz, T. Keller, and J. Li, *Adiabatic critical quantum metrology cannot reach the Heisenberg limit even when shortcuts to adiabaticity are applied*, Quantum **5**, 489 (2021).

[7] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, *Mastering the game of Go with deep neural networks and tree search*, Nature **529**, 484–489 (2016).

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection.* In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, (2016).

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language Models are Few-Shot Learners.* In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and

H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., (2020).

[10] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, *Machine learning and the physical sciences*, Rev. Mod. Phys. **91**, 045002 (2019).

[11] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, *A high-bias, low-variance introduction to Machine Learning for physicists*, Physics Reports **810**, 1–124 (2019).

[12] J. Carrasquilla, *Machine learning for quantum matter*, Advances in Physics: X **5**, 1797528 (2020).

[13] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, Nature Physics **13**, 431–434 (2016).

[14] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, *Learning phase transitions by confusion*, Nature Physics **13**, 435–439 (2017).

[15] G. Carleo and M. Troyer, *Solving the quantum many-body problem with artificial neural networks*, Science **355**, 602–606 (2017).

[16] 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).

[17] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, *Reinforcement Learning with Neural Networks for Quantum Feedback*, Phys. Rev. X **8**, 031084 (2018).

[18] K. Kottmann, P. Huembeli, M. Lewenstein, and A. Acín, *Unsupervised Phase Discovery with Deep Anomaly Detection*, Phys. Rev. Lett. **125**, 170603 (2020).

[19] P. Huembeli, A. Dauphin, P. Wittek, and C. Gogolin, *Automated discovery of characteristic features of phase transitions in many-body localization*, Phys. Rev. B **99**, 104106 (2019).

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

[21] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, *Neural-network quantum state tomography*, Nature Physics **14**, 447–450 (2018).

[22] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, *Optimizing Quantum Error Correction Codes with Reinforcement Learning*, Quantum **3**, 215 (2019).

[23] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, *Quantum error correction for the toric code using deep reinforcement learning*, Quantum **3**, 183 (2019).

[24] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, *Reinforcement learning decoders for fault-tolerant quantum computation*, Machine Learning: Science and Technology **2**, 025005 (2021).

[25] Y.-H. Zhang, P.-L. Zheng, Y. Zhang, and D.-L. Deng, *Topological Quantum Compiling with Reinforcement Learning*, Phys. Rev. Lett. **125**, 170501 (2020).

[26] L. Moro, M. G. A. Paris, M. Restelli, and E. Prati, *Quantum compiling by deep reinforcement learning*, Communications Physics **4**, 178 (2021).

[27] Z. He, L. Li, S. Zheng, Y. Li, and H. Situ, *Variational quantum compiling with double Q-learning*, New Journal of Physics **23**, 033002 (2021).

[28] T. Fösel, M. Yuezhen Niu, F. Marquardt, and L. Li, *Quantum circuit optimization with deep reinforcement learning*, arXiv e-prints , arXiv:2103.07585 (2021).

[29] K.-W. Zhao, W.-H. Kao, K.-H. Wu, and Y.-J. Kao, *Generation of ice states through deep reinforcement learning*, Phys. Rev. E **99**, 062106 (2019).

[30] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, *Universal quantum control through deep reinforcement learning*, npj Quantum Information **5**, 33 (2019).

[31] R.-H. He, R. Wang, S.-S. Nie, J. Wu, J.-H. Zhang, and Z.-M. Wang, *Deep reinforcement learning for universal quantum state preparation via dynamic pulse control*, EPJ Quantum Technology **8**, 29 (2021).

[32] A. Bolens and M. Heyl, *Reinforcement Learning for Digital Quantum Simulation*, Phys. Rev. Lett. **127**, 110502 (2021).

[33] E. Stoudenmire and D. J. Schwab. *Supervised Learning with Tensor Networks*. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., (2016).

[34] Z.-F. Gao, S. Cheng, R.-Q. He, Z. Y. Xie, H.-H. Zhao, Z.-Y. Lu, and T. Xiang, *Compressing deep neural networks by matrix product operators*, Phys. Rev. Research **2**, 023300 (2020).

[35] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, *An end-to-end trainable hybrid classical-quantum classifier*, Machine Learning: Science and Technology **2**, 045021 (2021).

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

[37] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics **349**, 117–158 (2014).

[38] J. Martyn, G. Vidal, C. Roberts, and S. Leichenauer, *Entanglement and Tensor Networks for Supervised Image Classification*, arXiv e-prints , arXiv:2007.06082 (2020).

[39] Y. Liu, W.-J. Li, X. Zhang, M. Lewenstein, G. Su, and S.-J. Ran, *Entanglement-Based Feature Extraction by Tensor Network Machine Learning*, Frontiers in Applied Mathematics and Statistics **7**, 716044 (2021).

[40] Z.-Z. Sun, C. Peng, D. Liu, S.-J. Ran, and G. Su, *Generative tensor network classification model for supervised machine learning*, Phys. Rev. B **101**, 075135 (2020).

[41] C. Guo, Z. Jie, W. Lu, and D. Poletti, *Matrix product operators for sequence-to-sequence learning*, Phys. Rev. E **98**, 042114 (2018).

[42] Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, *Unsupervised Generative Modeling Using Matrix Product States*, Phys. Rev. X **8**, 031012 (2018).

[43] J. Liu, S. Li, J. Zhang, and P. Zhang, *Tensor networks for unsupervised machine learning*, arXiv e-prints , arXiv:2106.12974 (2021).

[44] J. Stokes and J. Terilla, *Probabilistic Modeling with Matrix Product States*, Entropy **21**, 1236 (2019).

[45] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, *Variational quantum algorithms*, Nature Reviews Physics **3**, 625–644 (2021).

[46] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, *Noisy intermediate-scale quantum algorithms*, Rev. Mod. Phys. **94**, 015004 (2022).

[47] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, *Parameterized quantum circuits as machine learning models*, Quantum Science and Technology **4**, 043001 (2019).

[48] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, *The theory of variational hybrid quantum-classical algorithms*, New Journal of Physics **18**, 023023 (2015).

[49] J. Preskill, *Quantum Computing in the NISQ era and beyond*, Quantum **2**, 79 (2018).

[50] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, *Supervised learning with quantum-enhanced feature spaces*, Nature **567**, 209–212 (2019).

[51] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe, *Training of quantum circuits on a hybrid quantum computer*, Science Advances **5**, eaaw9918 (2019).

[52] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, and L. Sun, *Quantum generative adversarial learning in a superconducting quantum circuit*, Science Advances **5**, eaav2761 (2019).

[53] A. C. White, B. P. Anderson, and V. S. Bagnato, *Vortices and turbulence in trapped atomic condensates*, PNAS **111**, 4719–4726 (2014).

[54] R. Navarro, R. Carretero-González, P. J. Torres, P. G. Kevrekidis, D. J. Frantzeskakis, M. W. Ray, E. Altuntaş, and D. S. Hall, *Dynamics of a Few Corotating Vortices in Bose-Einstein Condensates*, Phys. Rev. Lett. **110**, 225301 (2013).

[55] T. Zhang, J. Schloss, A. Thomasen, L. J. O'Riordan, T. Busch, and A. White, *Chaotic few-body vortex dynamics in rotating Bose-Einstein condensates*, Phys. Rev. Fluids **4**, 054701 (2019).

[56] V. Koukouloyannis, G. Voyatzis, and P. G. Kevrekidis, *Dynamics of three noncorotating vortices in Bose-Einstein condensates*, Phys. Rev. E **89**, 042905 (2014).

[57] S. P. Johnstone, A. J. Groszek, P. T. Starkey, C. J. Billington, T. P. Simula, and K. Helmerson, *Evolution of large-scale flow from turbulence in a two-dimensional superfluid*, Science **364**, 1267–1271 (2019).

[58] K. E. Wilson, E. C. Samson, Z. L. Newman, T. W. Neely, and B. P. Anderson. *Experimental Methods for Generating Two-Dimensional Quantum Turbulence in Bose-Einstein Condensates*, pages 261–298. World Scientific, (2013).

[59] A. C. White, C. F. Barenghi, N. P. Proukakis, A. J. Youd, and D. H. Wacks, *Nonclassical Velocity Statistics in a Turbulent Atomic Bose-Einstein Condensate*, Phys. Rev. Lett. **104**, 075301 (2010).

[60] A. Rakonjac, A. L. Marchant, T. P. Billam, J. L. Helm, M. M. H. Yu, S. A. Gardiner, and S. L. Cornish, *Measuring the disorder of vortex lattices in a Bose-Einstein condensate*, Phys. Rev. A **93**, 013607 (2016).

[61] S. Sachdev, *Quantum Phase Transitions*, Cambridge University Press, 2 edition (2011).

[62] K. Kottmann, P. Corboz, M. Lewenstein, and A. Acín, *Unsupervised mapping of phase diagrams of 2D systems from infinite projected entangled-pair states via deep anomaly detection*, SciPost Phys. **11**, 025 (2021).

[63] N. Käming, A. Dawid, K. Kottmann, M. Lewenstein, K. Sengstock, A. Dauphin, and C. Weitenberg, *Unsupervised machine learning of topological phase transitions from experimental data*, Machine Learning: Science and Technology **2**, 035037 (2021).

[64] D. Contessi, E. Ricci, A. Recati, and M. Rizzi, *Detection of Berezinskii-Kosterlitz-Thouless transition via Generative Adversarial Networks*, SciPost Phys. **12**, 107 (2022).

[65] Y.-H. Tsai, K.-F. Chiu, Y.-C. Lai, K.-J. Su, T.-P. Yang, T.-P. Cheng, G.-Y. Huang, and M.-C. Chung, *Deep learning of topological phase transitions from entanglement aspects: An unsupervised way*, Phys. Rev. B **104**, 165108 (2021).

[66] H. C. Fogedby, *The Ising chain in a skew magnetic field*, Journal of Physics C: Solid State Physics **11**, 2801–2813 (1978).

[67] A. A. Ovchinnikov, D. V. Dmitriev, V. Y. Krivnov, and V. O. Cheranovskii, *Antiferromagnetic Ising chain in a mixed transverse and longitudinal magnetic field*, Phys. Rev. B **68**, 214406 (2003).

[68] K. Sugimoto, S. Ejima, F. Lange, and H. Fehske, *Quantum phase transitions in the dimerized extended Bose-Hubbard model*, Phys. Rev. A **99**, 012122 (2019).

[69] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, *Quantum entanglement*, Rev. Mod. Phys. **81**, 865–942 (2009).

[70] A. M. Childs, *Universal Computation by Quantum Walk*, Phys. Rev. Lett. **102**, 180501 (2009).

[71] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon, *Universal quantum computation using the discrete-time quantum walk*, Phys. Rev. A **81**, 042330 (2010).

[72] S. Srikara and C. M. Chandrashekar, *Quantum direct communication protocols using discrete-time quantum walk*, Quantum Inf. Process. **19**, 295 (2020).

[73] Y. Shang, Y. Wang, M. Li, and R. Lu, *Quantum communication protocols by quantum walks with two coins*, EPL (Europhysics Letters) **124**, 60009 (2019).

[74] Y. Wang, Y. Shang, and P. Xue, *Generalized teleportation by quantum walks*, Quantum Information Processing **16**, 1–13 (2017).

[75] Y. Chatterjee, V. Devrari, B. K. Behera, and P. K. Panigrahi, *Experimental realization of quantum teleportation using coined quantum walks*, Quantum Information Processing **19**, 31 (2019).

[76] A. Acín, I. Bloch, H. Buhrman, T. Calarco, C. Eichler, J. Eisert, D. Esteve, N. Gisin, S. J. Glaser, F. Jelezko, S. Kuhr, M. Lewenstein, M. F. Riedel, P. O. Schmidt, R. Thew, A. Wallraff, I. Walmsley, and F. K. Wilhelm, *The quantum technologies roadmap: a European community view*, New Journal of Physics **20**, 080201 (2018).

[77] M. Lewenstein, A. Sanpera, V. Ahufinger, B. Damski, A. Sen(De), and U. Sen, *Ultracold atomic gases in optical lattices: mimicking condensed matter physics and beyond*, Advances in Physics **56**, 243–379 (2007).

[78] R. Blatt and C. Roos, *Quantum Simulations with Trapped Ions*, Nature Physics **8**, 277–284 (2012).

[79] F. Casola, T. van der Sar, and A. Yacoby, *Probing condensed matter physics with magnetometry based on nitrogen-vacancy centres in diamond*, Nature Reviews Materials **3**, 17088 (2018).

[80] J. L. O'Brien, A. Furusawa, and J. Vučković, *Photonic quantum technologies*, Nature Photonics **3**, 687–695 (2009).

[81] A. Dawid, J. Arnold, B. Requena, A. Gresch, M. Płodzień, K. Donatella, K. A. Nicoli, P. Stornati, R. Koch, M. Büttner, R. Okuła, G. Muñoz-Gil, R. A. Vargas-Hernández, A. Cervera-Lierta, J. Carrasquilla, V. Dunjko, M. Gabrié, P. Huembeli, E. van Nieuwenburg, F. Vicentini, L. Wang, S. J. Wetzel, G. Carleo, E. Greplová, R. Krems, F. Marquardt, M. Tomza, M. Lewenstein, and A. Dauphin, *Modern applications of machine learning in quantum sciences*, arXiv e-prints , arXiv:2204.04198 (2022).

[82] M. Krenn, J. Landgraf, T. Foesel, and F. Marquardt, *Artificial Intelligence and Machine Learning for Quantum Technologies*, arXiv e-prints , arXiv:2208.03836 (2022).

[83] V. Dunjko and H. J. Briegel, *Machine learning & artificial intelligence in the quantum domain: a review of recent progress*, Rep. Prog. Phys. **81**, 074001 (2018).

[84] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-Resolution Image Synthesis with Latent Diffusion Models*, arXiv e-prints , arXiv:2112.10752 (2021).

[85] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, *Zero-Shot Text-to-Image Generation*, arXiv e-prints , arXiv:2102.12092 (2021).

[86] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, second edition (2018).

[87] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, *Human-level control through deep reinforcement learning*, Nature **518**, 529–33 (2015).

[88] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver, *Grandmaster level in StarCraft II using multi-agent reinforcement learning*, Nature **575**, 350–354 (2019).

[89] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press (2015).

[90] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press (2016). `http://www.deeplearningbook.org`.

[91] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg (2006).

[92] Y. LeCun, Y. Bengio, and G. Hinton, *Deep Learning*, Nature **521**, 436–44 (2015).

[93] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61**, 85–117 (2015).

[94] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks **4**, 251–257 (1991).

[95] G. V. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals and Systems **2**, 303–314 (1989).

[96] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature **323**, 533–536 (1986).

[97] W. Baur and V. Strassen, *The complexity of partial derivatives*, Theoretical Computer Science **22**, 317–330 (1983).

[98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86**, 2278–2324 (1998).

[99] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press (2010).

[100] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., (2012).

[101] G. E. Hinton and R. Zemel. *Autoencoders, Minimum Description Length and Helmholtz Free Energy.* In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, (1993).

[102] G. E. Hinton and R. R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, Science **313**, 504–507 (2006).

[103] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, *Anomaly Detection Using Autoencoders in High Performance Computing Systems*, Proceedings of the AAAI Conference on Artificial Intelligence **33**, 9428–9433 (2019).

[104] J. Xie, L. Xu, and E. Chen. *Image Denoising and Inpainting with Deep Neural Networks.* In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., (2012).

[105] R. Zhang, P. Isola, and A. A. Efros, *Colorful Image Colorization*, arXiv e-prints , arXiv:1603.08511 (2016).

[106] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, arXiv e-prints , arXiv:1312.6114 (2013).

[107] J. C. Bridgeman and C. T. Chubb, *Hand-waving and interpretive dance: an introductory course on tensor networks*, Journal of Physics A: Mathematical and Theoretical **50**, 223001 (2017).

[108] J. Hauschild and F. Pollmann, *Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy)*, SciPost Phys. Lect. Notes **5** (2018).

[109] R. Orús, *Tensor networks for complex quantum systems*, Nature Reviews Physics **1**, 538–550 (2019).

[110] G. Evenbly, *A Practical Guide to the Numerical Implementation of Tensor Networks I: Contractions, Decompositions, and Gauge Freedom*, Frontiers in Applied Mathematics and Statistics **8**, 806549 (2022).

[111] L. Chi-Chung, P. Sadayappan, and R. Wenger, *On Optimizing a Class of Multi-Dimensional Loops with Reduction for Parallel Execution*, Parallel Processing Letters **07**, 157–168 (1997).

[112] I. Arad and Z. Landau, *Quantum Computation and the Evaluation of Tensor Networks*, SIAM Journal on Computing **39**, 3089–3121 (2010).

[113] S. Östlund and S. Rommer, *Thermodynamic Limit of Density Matrix Renormalization*, Phys. Rev. Lett. **75**, 3537–3540 (1995).

[114] S. Rommer and S. Östlund, *Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group*, Phys. Rev. B **55**, 2164–2181 (1997).

[115] M. Fannes, B. Nachtergaele, and R. F. Werner, *Finitely correlated states on quantum spin chains*, Communications in Mathematical Physics **144**, 443 – 490 (1992).

[116] I. P. McCulloch, *From density-matrix renormalization group to matrix product states*, Journal of Statistical Mechanics: Theory and Experiment **2007**, P10014–P10014 (2007).

[117] Y.-Y. Shi, L.-M. Duan, and G. Vidal, *Classical simulation of quantum many-body systems with a tree tensor network*, Phys. Rev. A **74**, 022320 (2006).

[118] G. Vidal, *Entanglement Renormalization*, Phys. Rev. Lett. **99**, 220405 (2007).

[119] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization*, Phys. Rev. B **79**, 144108 (2009).

[120] A. Klumper, A. Schadschneider, and J. Zittartz, *Equivalence and solution of anisotropic spin-1 models and generalized t-J fermion models in one dimension*, Journal of Physics A: Mathematical and General **24**, L955–L959 (1991).

[121] I. V. Oseledets, *Tensor-Train Decomposition*, SIAM Journal on Scientific Computing **33**, 2295–2317 (2011).

[122] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, *Matrix product states and projected entangled pair states: Concepts, symmetries, theorems*, Rev. Mod. Phys. **93**, 045003 (2021).

[123] M. B. Hastings, *An area law for one-dimensional quantum systems*, Journal of Statistical Mechanics: Theory and Experiment **2007**, P08024–P08024 (2007).

[124] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Entropy Scaling and Simulability by Matrix Product States*, Phys. Rev. Lett. **100**, 030504 (2008).

[125] M. B. Hastings, *Solving gapped Hamiltonians locally*, Phys. Rev. B **73**, 085115 (2006).

[126] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett. **69**, 2863–2866 (1992).

[127] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Pižorn, H. Verschelde, and F. Verstraete, *Time-Dependent Variational Principle for Quantum Lattices*, Phys. Rev. Lett. **107**, 070601 (2011).

[128] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, *Time-evolution methods for matrix-product states*, Annals of Physics **411**, 167998 (2019).

[129] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete, *Matrix product operator representations*, New Journal of Physics **12**, 025012 (2010).

[130] F. Verstraete and J. I. Cirac, *Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions*, arXiv e-prints , arXiv:cond–mat/0407066 (2004).

[131] J. Wang, C. Roberts, G. Vidal, and S. Leichenauer, *Anomaly Detection with Tensor Networks*, arXiv e-prints , arXiv:2006.02516 (2020).

[132] Z.-Z. Sun, S.-J. Ran, and G. Su, *Tangent-space gradient optimization of tensor network for machine learning*, Phys. Rev. E **102**, 012152 (2020).

[133] Z. Liu, L.-W. Yu, L. M. Duan, and D.-L. Deng, *The Presence and Absence of Barren Plateaus in Tensor-network Based Machine Learning*, arXiv e-prints , arXiv:2108.08312 (2021).

[134] E. Cervero Martín, K. Plekhanov, and M. Lubasch, *Barren plateaus in quantum tensor network optimization*, arXiv e-prints , arXiv:2209.00292 (2022).

[135] S. Yen-Chi Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, *Hybrid quantum-classical classifier based on tensor network and variational quantum circuit*, arXiv e-prints , arXiv:2011.14651 (2020).

[136] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, and Y.-J. Kao, *Variational quantum reinforcement learning via evolutionary optimization*, Machine Learning: Science and Technology **3**, 015025 (2022).

[137] S. Cheng, L. Wang, and P. Zhang, *Supervised learning with projected entangled pair states*, Phys. Rev. B **103**, 125117 (2021).

[138] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su, and M. Lewenstein, *Machine learning by unitary tensor network of hierarchical tree structure*, New Journal of Physics **21**, 073059 (2019).

[139] J. A. Reyes and E. M. Stoudenmire, *Multi-scale tensor network architecture for machine learning*, Machine Learning: Science and Technology **2**, 035036 (2021).

[140] E. M. Stoudenmire, *Learning relevant features of data with multi-scale tensor networks*, Quantum Science and Technology **3**, 034003 (2018).

[141] S. Cheng, L. Wang, T. Xiang, and P. Zhang, *Tree tensor networks for generative modeling*, Phys. Rev. B **99**, 155131 (2019).

[142] M. L. Wall and G. D'Aguanno, *Tree-tensor-network classifiers for machine learning: From quantum inspired to quantum assisted*, Phys. Rev. A **104**, 042408 (2021).

[143] I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and I. Cirac. *Expressive power of tensor-network factorizations for probabilistic modeling.* In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., (2019).

[144] I. Glasser, N. Pancotti, and J. I. Cirac, *From probabilistic graphical models to generalized tensor networks for supervised learning*, arXiv e-prints , arXiv:1806.05964 (2018).

[145] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, *Quantum Entanglement in Deep Learning Architectures*, Phys. Rev. Lett. **122**, 065301 (2019).

[146] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, *Neural-Network Quantum States, String-Bond States, and Chiral Topological States*, Phys. Rev. X **8**, 011006 (2018).

[147] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, *Equivalence of restricted Boltzmann machines and tensor network states*, Phys. Rev. B **97**, 085104 (2018).

[148] M. Schuld, I. Sinayskiy, and F. Petruccione, *An introduction to quantum machine learning*, Contemporary Physics **56**, 172–185 (2015).

[149] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum machine learning*, Nature **549**, 195–202 (2017).

[150] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, Springer Publishing Company, Incorporated, 1st edition (2018).

[151] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, *Quantum machine learning: a classical perspective*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**, 20170551 (2018).

[152] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, *Experimental realization of any discrete unitary operator*, Phys. Rev. Lett. **73**, 58–61 (1994).

[153] D. P. DiVincenzo, *Two-bit gates are universal for quantum computation*, Phys. Rev. A **51**, 1015–1022 (1995).

[154] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, *Quantum gradient descent and Newton's method for constrained polynomial optimization*, New Journal of Physics **21**, 073023 (2019).

[155] S. Lloyd, M. Mohseni, and P. Rebentrost, *Quantum principal component analysis*, Nature Physics **10**, 631–633 (2014).

[156] P. Rebentrost, M. Mohseni, and S. Lloyd, *Quantum Support Vector Machine for Big Data Classification*, Phys. Rev. Lett. **113**, 130503 (2014).

[157] F. Tacchino, C. Macchiavello, D. Gerace, and D. Bajoni, *An artificial neuron implemented on an actual quantum processor*, npj Quantum Information **5**, 26 (2019).

[158] E. Torrontegui and J. J. García-Ripoll, *Unitary quantum perceptron as efficient universal approximator*, Europhysics Letters **125**, 30004 (2019).

[159] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, *Quantum generalisation of feedforward neural networks*, npj Quantum Information **3**, 36 (2017).

[160] F. Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, and D. Bajoni, *Quantum implementation of an artificial feed-forward neural network*, Quantum Science and Technology **5**, 044010 (2020).

[161] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, *Quantum Boltzmann Machine*, Phys. Rev. X **8**, 021050 (2018).

[162] M. Schuld and N. Killoran, *Quantum Machine Learning in Feature Hilbert Spaces*, Phys. Rev. Lett. **122**, 040504 (2019).

[163] R. Mengoni and A. Di Pierro, *Kernel methods in Quantum Machine Learning*, Quantum Machine Intelligence **1**, pages 65–71 (2019).

[164] T. Hofmann, B. Schölkopf, and A. J. Smola, *Kernel methods in machine learning*, The Annals of Statistics **36**, 1171 – 1220 (2008).

[165] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, *Quantum circuit learning*, Phys. Rev. A **98**, 032309 (2018).

[166] E. Farhi and H. Neven, *Classification with Quantum Neural Networks on Near Term Processors*, arXiv e-prints , arXiv:1802.06002 (2018).

[167] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature **549**, 242–246 (2017).

[168] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, *Quantum embeddings for machine learning*, arXiv e-prints , arXiv:2001.03622 (2020).

[169] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, *Data re-uploading for a universal quantum classifier*, Quantum **4**, 226 (2020).

[170] M. Schuld, R. Sweke, and J. J. Meyer, *Effect of data encoding on the expressive power of variational quantum-machine-learning models*, Phys. Rev. A **103**, 032430 (2021).

[171] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, *Evaluating analytic gradients on quantum hardware*, Phys. Rev. A **99**, 032331 (2019).

[172] J. Spall, *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE Transactions on Automatic Control **37**, 332–341 (1992).

[173] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Barren plateaus in quantum neural network training landscapes*, Nature Communications **9**, 4812 (2018).

[174] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, *Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits*, Nature Communications **12**, 1791 (2020).

[175] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib, *Layerwise learning for quantum neural networks*, Quantum Machine Intelligence **3**, 5 (2021).

[176] T. Haug and M. S. Kim, *Optimal training of variational quantum algorithms without barren plateaus*, arXiv e-prints , arXiv:2104.14543 (2021).

[177] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum autoencoders for efficient compression of quantum data*, Quantum Science and Technology **2**, 045001 (2017).

[178] C. Bravo-Prieto, *Quantum autoencoders with enhanced data encoding*, Machine Learning: Science and Technology **2**, 035028 (2021).

[179] A. Pepper, N. Tischler, and G. J. Pryde, *Experimental Realization of a Quantum Autoencoder: The Compression of Qutrits via Machine Learning*, Phys. Rev. Lett. **122**, 060501 (2019).

[180] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, *Machine Learning Phases of Strongly Correlated Fermions*, Phys. Rev. X **7**, 031038 (2017).

[181] D. Carvalho, N. A. García-Martínez, J. L. Lado, and J. Fernández-Rossier, *Real-space mapping of topological invariants using artificial neural networks*, Phys. Rev. B **97**, 115453 (2018).

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

[183] Y. Zhang, R. G. Melko, and E.-A. Kim, *Machine learning $\mathbb{Z}_2$ quantum spin liquids with quasiparticle statistics*, Phys. Rev. B **96**, 245119 (2017).

[184] Y.-T. Hsu, X. Li, D.-L. Deng, and S. Das Sarma, *Machine Learning Many-Body Localization: Search for the Elusive Nonergodic Metal*, Phys. Rev. Lett. **121**, 245701 (2018).

[185] J. Venderley, V. Khemani, and E.-A. Kim, *Machine Learning Out-of-Equilibrium Phases of Matter*, Phys. Rev. Lett. **120**, 257204 (2018).

[186] L. Wang, *Discovering phase transitions with unsupervised learning*, Phys. Rev. B **94**, 195105 (2016).

[187] J. F. Rodriguez-Nieva and M. S. Scheurer, *Identifying topological order through unsupervised machine learning*, Nature Physics **15**, 790–795 (2019).

[188] S. Acevedo, M. Arlego, and C. A. Lamas, *Phase diagram study of a two-dimensional frustrated antiferromagnet via unsupervised machine learning*, Phys. Rev. B **103**, 134422 (2021).

[189] O. Sharir, A. Shashua, and G. Carleo, *Neural tensor contractions and the expressive power of deep neural quantum states*, arXiv e-prints , arXiv:2103.10293 (2021).

[190] D.-L. Deng, X. Li, and S. Das Sarma, *Quantum Entanglement in Neural Network States*, Phys. Rev. X **7**, 021021 (2017).

[191] X. Gao and L.-M. Duan, *Efficient Representation of Quantum Many-body States with Deep Neural Networks*, Nature Communications **8**, 662 (2017).

[192] S. Lu, X. Gao, and L.-M. Duan, *Efficient representation of topologically ordered states with restricted Boltzmann machines*, Phys. Rev. B **99**, 155136 (2019).

[193] D.-L. Deng, X. Li, and S. Das Sarma, *Machine learning topological states*, Phys. Rev. B **96**, 195145 (2017).

[194] X. Liang, W.-Y. Liu, P.-Z. Lin, G.-C. Guo, Y.-S. Zhang, and L. He, *Solving frustrated quantum many-particle models with convolutional neural networks*, Phys. Rev. B **98**, 104426 (2018).

[195] K. Choo, T. Neupert, and G. Carleo, *Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states*, Phys. Rev. B **100**, 125124 (2019).

[196] K. Choo, G. Carleo, N. Regnault, and T. Neupert, *Symmetries and Many-Body Excitations with Neural-Network Quantum States*, Phys. Rev. Lett. **121**, 167204 (2018).

[197] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, *Restricted Boltzmann machine learning for solving strongly correlated quantum systems*, Phys. Rev. B **96**, 205152 (2017).

[198] N. Yoshioka and R. Hamazaki, *Constructing neural stationary states for open quantum many-body systems*, Phys. Rev. B **99**, 214306 (2019).

[199] M. J. Hartmann and G. Carleo, *Neural-Network Approach to Dissipative Quantum Many-Body Dynamics*, Phys. Rev. Lett. **122**, 250502 (2019).

[200] A. Nagy and V. Savona, *Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems*, Phys. Rev. Lett. **122**, 250501 (2019).

[201] F. Vicentini, A. Biella, N. Regnault, and C. Ciuti, *Variational Neural-Network Ansatz for Steady States in Open Quantum Systems*, Phys. Rev. Lett. **122**, 250503 (2019).

[202] M. Schmitt and M. Heyl, *Quantum Many-Body Dynamics in Two Dimensions with Artificial Neural Networks*, Phys. Rev. Lett. **125**, 100503 (2020).

[203] S. Czischek, M. Gärttner, and T. Gasenzer, *Quenches near Ising quantum criticality as a challenge for artificial neural networks*, Phys. Rev. B **98**, 024311 (2018).

[204] G. Torlai, B. Timar, E. P. L. van Nieuwenburg, H. Levine, A. Omran, A. Keesling, H. Bernien, M. Greiner, V. Vuletić, M. D. Lukin, R. G. Melko, and M. Endres, *Integrating Neural Networks with a Quantum Simulator for State Reconstruction*, Phys. Rev. Lett. **123**, 230504 (2019).

[205] M. Bukov, *Reinforcement learning for autonomous preparation of Floquet-engineered states: Inverting the quantum Kapitza oscillator*, Phys. Rev. B **98**, 224305 (2018).

[206] T. Haug, W.-K. Mok, J.-B. You, W. Zhang, C. E. Png, and L.-C. Kwek, *Classifying global state preparation via deep reinforcement learning*, Machine Learning: Science and Technology **2**, 01LT02 (2020).

[207] J. Mackeprang, D. B. R. Dasari, and J. Wrachtrup, *A reinforcement learning approach for quantum state engineering*, Quantum Machine Intelligence **2**, 5 (2020).

[208] J. Yao, M. Bukov, and L. Lin. *Policy Gradient based Quantum Approximate Optimization Algorithm.* In J. Lu and R. Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 605–634. PMLR, (2020).

[209] J. Yao, P. Köttering, H. Gundlach, L. Lin, and M. Bukov, *Noise-Robust End-to-End Quantum Control using Deep Autoregressive Policy Networks*, arXiv e-prints , arXiv:2012.06701 (2020).

[210] T. Haug, R. Dumke, L.-C. Kwek, C. Miniatura, and L. Amico, *Machine-learning engineering of quantum currents*, Phys. Rev. Research **3**, 013034 (2021).

[211] J. Yao, L. Lin, and M. Bukov, *Reinforcement Learning for Many-Body Ground-State Preparation Inspired by Counterdiabatic Driving*, Phys. Rev. X **11**, 031070 (2021).

[212] S.-F. Guo, F. Chen, Q. Liu, M. Xue, J.-J. Chen, J.-H. Cao, T.-W. Mao, M. K. Tey, and L. You, *Faster State Preparation across Quantum Phase Transition Assisted by Reinforcement Learning*, Phys. Rev. Lett. **126**, 060401 (2021).

[213] C. Cao, Z. An, S.-Y. Hou, D. L. Zhou, and B. Zeng, *Quantum imaginary time evolution steered by reinforcement learning*, Communications Physics **5**, 57 (2022).

[214] H. Xu, J. Li, L. Liu, Y. Wang, H. Yuan, and X. Wang, *Generalizable control for quantum parameter estimation through reinforcement learning*, npj Quantum Information **5**, 82 (2019).

[215] J. Schuff, L. J. Fiderer, and D. Braun, *Improving the dynamics of quantum sensors with reinforcement learning*, New Journal of Physics **22**, 035001 (2020).

[216] J. R. Dorfman, *An Introduction to Chaos in Nonequilibrium Statistical Mechanics*, Cambridge University Press (1999).

[217] V. Špička, P. D. Keefe, and T. M. Nieuwenhuizen, *Non-equilibrium dynamics: quantum systems and foundations of quantum mechanics*, EPJ ST **227**, 1837–1848 (2019).

[218] F. Dalfovo, S. Giorgini, L. P. Pitaevskii, and S. Stringari, *Theory of Bose-Einstein condensation in trapped gases*, Rev. Mod. Phys. **71**, 463–512 (1999).

[219] A. L. Fetter and A. A. Svidzinsky, *Vortices in a trapped dilute Bose-Einstein condensate*, J. Phys. Condens. Matter **13**, R135–R194 (2001).

[220] N. G. Parker, B. Jackson, A. M. Martin, and C. S. Adams. *Vortices in Bose-Einstein Condensates: Theory*, pages 173–189. Springer Berlin Heidelberg, (2008).

[221] F. Chevy. *Vortices in Bose-Einstein Condensates: Experiments*, pages 191–207. Springer Berlin Heidelberg, (2008).

[222] S. Inouye, S. Gupta, T. Rosenband, A. P. Chikkatur, A. Görlitz, T. L. Gustavson, A. E. Leanhardt, D. E. Pritchard, and W. Ketterle, *Observation of Vortex Phase Singularities in Bose-Einstein Condensates*, Phys. Rev. Lett. **87**, 080402 (2001).

[223] B. P. Anderson, P. C. Haljan, C. A. Regal, D. L. Feder, L. A. Collins, C. W. Clark, and E. A. Cornell, *Watching Dark Solitons Decay into Vortex Rings in a Bose-Einstein Condensate*, Phys. Rev. Lett. **86**, 2926–2929 (2001).

[224] J. R. Abo-Shaeer, C. Raman, J. M. Vogels, and W. Ketterle, *Observation of Vortex Lattices in Bose-Einstein Condensates*, Science **292**, 476–479 (2001).

[225] D. V. Freilich, D. M. Bianchi, A. M. Kaufman, T. K. Langin, and D. S. Hall, *Real-Time Dynamics of Single Vortex Lines and Vortex Dipoles in a Bose-Einstein Condensate*, Science **329**, 1182–1185 (2010).

[226] K. E. Wilson, Z. L. Newman, J. D. Lowney, and B. P. Anderson, *In situ imaging of vortices in Bose-Einstein condensates*, Phys. Rev. A **91**, 023621 (2015).

[227] P. C. Haljan, I. Coddington, P. Engels, and E. A. Cornell, *Driving Bose-Einstein-Condensate Vorticity with a Rotating Normal Cloud*, Phys. Rev. Lett. **87**, 210403 (2001).

[228] S. Serafini, L. Galantucci, E. Iseni, T. Bienaimé, R. N. Bisset, C. F. Barenghi, F. Dalfovo, G. Lamporesi, and G. Ferrari, *Vortex Reconnections and Rebounds in Trapped Atomic Bose-Einstein Condensates*, Phys. Rev. X **7**, 021031 (2017).

[229] W. J. Kwon, G. Moon, J.-y. Choi, S. W. Seo, and Y.-i. Shin, *Relaxation of superfluid turbulence in highly oblate Bose-Einstein condensates*, Phys. Rev. A **90**, 063627 (2014).

[230] T. W. Neely, A. S. Bradley, E. C. Samson, S. J. Rooney, E. M. Wright, K. J. H. Law, R. Carretero-González, P. G. Kevrekidis, M. J. Davis, and B. P. Anderson, *Characteristics of Two-Dimensional Quantum Turbulence in a Compressible Superfluid*, Phys. Rev. Lett. **111**, 235301 (2013).

[231] M. T. Reeves, K. Goddard-Lee, G. Gauthier, O. R. Stockdale, H. Salman, T. Edmonds, X. Yu, A. S. Bradley, M. Baker, H. Rubinsztein-Dunlop, M. J. Davis, and T. W. Neely, *Turbulent Relaxation to Equilibrium in a Two-Dimensional Quantum Vortex Gas*, Phys. Rev. X **12**, 011031 (2022).

[232] G. Gauthier, M. T. Reeves, X. Yu, A. S. Bradley, M. A. Baker, T. A. Bell, H. Rubinsztein-Dunlop, M. J. Davis, and T. W. Neely, *Giant vortex clusters in a two-dimensional quantum fluid*, Science **364**, 1264–1267 (2019).

[233] W. J. Kwon, J. H. Kim, S. W. Seo, and Y. Shin, *Observation of von Kármán Vortex Street in an Atomic Superfluid Gas*, Phys. Rev. Lett. **117**, 245301 (2016).

[234] S. W. Seo, B. Ko, J. H. Kim, and Y. Shin, *Observation of vortex-antivortex pairing in decaying 2D turbulence of a superfluid gas*, Sci. Rep. **7**, 4587 (2017).

[235] A. Aftalion and Q. Du, *Vortices in a rotating Bose-Einstein condensate: Critical angular velocities and energy diagrams in the Thomas-Fermi regime*, Phys. Rev. A **64**, 063603 (2001).

[236] A. B. Ortega, S. Bucio-Pacheco, S. Lopez-Huidobro, L. Perez-Garcia, F. J. Poveda-Cuevas, J. A. Seman, A. V. Arzola, and K. Volke-Sepúlveda, *Creation of optical speckle by randomizing a vortex-lattice*, Opt. Express **27**, 4105–4115 (2019).

[237] A. J. Groszek, M. J. Davis, and T. P. Simula, *Decaying quantum turbulence in a two-dimensional Bose-Einstein condensate at finite temperature*, SciPost Phys. **8**, 39 (2020).

[238] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*, Wiley Publishing (2009).

[239] A. J. Barker, H. Style, K. Luksch, S. Sunami, D. Garrick, F. Hill, C. J. Foot, and E. Bentine, *Applying machine learning optimization methods to the production of a quantum gas*, MLST **1**, 015007 (2020).

[240] H. Saito, *Creation and Manipulation of Quantized Vortices in Bose–Einstein Condensates Using Reinforcement Learning*, J. Phys. Soc. Jpn. **89**, 074006 (2020).

[241] Y. Xiao, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du, and X. Lan, *A review of object detection based on deep learning*, Multimed. Tools. Appl. **79**, 23729–23791 (2020).

[242] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, *Deep Learning for Generic Object Detection: A Survey*, International Journal of Computer Vision **128**, 261–318 (2020).

[243] E. N. Minor, S. D. Howard, A. A. S. Green, M. A. Glaser, C. S. Park, and N. A. Clark, *End-to-end machine learning for experimental physics: using simulated data to train a neural network for object detection in video microscopy*, Soft Matter **16**, 1751–1759 (2020).

[244] M. Usman, Y. Z. Wong, C. D. Hill, and L. C. L. Hollenberg, *Framework for atomic-level characterisation of quantum computer arrays by machine learning*, NPJ Comput. Mater. **6**, 19 (2020).

[245] L. R. Hofer, M. Krstajić, P. Juhász, A. L. Marchant, and R. P. Smith, *Atom cloud detection and segmentation using a deep neural network*, Machine Learning: Science and Technology **2**, 045008 (2021).

[246] S. Guo, A. R. Fritsch, C. Greenberg, I. B. Spielman, and J. P. Zwolak, *Machine-learning enhanced dark soliton detection in Bose–Einstein condensates*, Machine Learning: Science and Technology **2**, 035020 (2021).

[247] Y. Luo, Y. Shao, H. Chu, B. Wu, M. Huang, and Y. Rao. *CNN-based blade tip vortex region detection in flow field.* In *Eleventh International Conference on Graphics and Image Processing*, volume 11373, pages 182 – 187. SPIE, (2020).

[248] X. Bai, C. Wang, and C. Li, *A Streampath-Based RCNN Approach to Ocean Eddy Detection*, IEEE Access **7**, 106336–106345 (2019).

[249] L. Salasnich, A. Parola, and L. Reatto, *Effective wave equations for the dynamics of cigar-shaped and disk-shaped Bose condensates*, Phys. Rev. A **65**, 043614 (2002).

[250] D. S. Petrov, M. Holzmann, and G. V. Shlyapnikov, *Bose-Einstein Condensation in Quasi-2D Trapped Gases*, Phys. Rev. Lett. **84**, 2551–2555 (2000).

[251] W.-M. Liu and E. Kengne. *Overview of Nonlinear Schrödinger Equations*, pages 1–13. Springer Singapore, (2019).

[252] E. Lundh, C. J. Pethick, and H. Smith, *Zero-temperature properties of a trapped Bose-condensed gas: Beyond the Thomas-Fermi approximation*, Phys. Rev. A **55**, 2126–2131 (1997).

[253] F. Chevy, K. W. Madison, and J. Dalibard, *Measurement of the Angular Momentum of a Rotating Bose-Einstein Condensate*, Phys. Rev. Lett. **85**, 2223–2227 (2000).

[254] K. W. Madison, F. Chevy, W. Wohlleben, and J. Dalibard, *Vortex Formation in a Stirred Bose-Einstein Condensate*, Phys. Rev. Lett. **84**, 806–809 (2000).

[255] M. Tsubota, K. Kasamatsu, and M. Ueda, *Vortex lattice formation in a rotating Bose-Einstein condensate*, Phys. Rev. A **65**, 023603 (2002).

[256] K. Sasaki, N. Suzuki, and H. Saito, *Bénard–von Kármán Vortex Street in a Bose-Einstein Condensate*, Phys. Rev. Lett. **104**, 150404 (2010).

[257] T. W. Neely, E. C. Samson, A. S. Bradley, M. J. Davis, and B. P. Anderson, *Observation of Vortex Dipoles in an Oblate Bose-Einstein Condensate*, Phys. Rev. Lett. **104**, 160401 (2010).

[258] L. J. O'Riordan and T. Busch, *Topological defect dynamics of vortex lattices in Bose-Einstein condensates*, Phys. Rev. A **94**, 053603 (2016).

[259] A. E. Leanhardt, A. Görlitz, A. P. Chikkatur, D. Kielpinski, Y. Shin, D. E. Pritchard, and W. Ketterle, *Imprinting Vortices in a Bose-Einstein Condensate using Topological Phases*, Phys. Rev. Lett. **89**, 190403 (2002).

[260] L. Dobrek, M. Gajda, M. Lewenstein, K. Sengstock, G. Birkl, and W. Ertmer, *Optical generation of vortices in trapped Bose-Einstein condensates*, Phys. Rev. A **60**, R3381–R3384 (1999).

[261] M. F. Andersen, C. Ryu, P. Cladé, V. Natarajan, A. Vaziri, K. Helmerson, and W. D. Phillips, *Quantized Rotation of Atoms from Photons with Orbital Angular Momentum*, Phys. Rev. Lett. **97**, 170406 (2006).

[262] X. Zhou, D. Wang, and P. Krähenbühl, *Objects as Points*, arXiv e-prints , arXiv:1904.07850 (2019).

[263] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization.* In *3rd International Conference on Learning Representations, San Diego, 2015*, (2015).

[264] F. Metz. *Deep learning based quantum vortex detection in atomic Bose-Einstein condensates.* `https://github.com/frmetz/quantum_vortex_detection`, (2020).

[265] J. Javanainen and J. Ruostekoski, *Symbolic calculation in development of algorithms: split-step methods for the Gross–Pitaevskii equation*, J Phys A Math Gen **39**, L179–L184 (2006).

[266] K. Kasamatsu, M. Tsubota, and M. Ueda, *Nonlinear dynamics of vortex lattice formation in a rotating Bose-Einstein condensate*, Phys. Rev. A **67**, 033610 (2003).

[267] Z. Yang, Y. Liu, L. Liu, X. Tang, J. Xie, and X. Gao, *Detecting Small Objects in Urban Settings Using SlimNet Model*, IEEE Trans. Geosci. Remote Sens. **57**, 8445–8457 (2019).

[268] M. Innes, *Flux: Elegant machine learning with Julia*, J. Open Source Softw. **3**, 602 (2018).

[269] S. Fölling, F. Gerbier, A. Widera, O. Mandel, T. Gericke, and I. Bloch, *Spatial quantum noise interferometry in expanding ultracold atom clouds*, Nature **434**, 481–484 (2005).

[270] G. Ness, A. Vainbaum, C. Shkedrov, Y. Florshaim, and Y. Sagi, *Single-Exposure Absorption Imaging of Ultracold Atoms Using Deep Learning*, Phys. Rev. Applied **14**, 014011 (2020).

[271] B. Song, C. He, Z. Ren, E. Zhao, J. Lee, and G.-B. Jo, *Effective Statistical Fringe Removal Algorithm for High-Sensitivity Imaging of Ultracold Atoms*, Phys. Rev. Applied **14**, 034006 (2020).

[272] S. Gautam, A. Roy, and S. Mukerjee, *Finite-temperature dynamics of vortices in Bose-Einstein condensates*, Phys. Rev. A **89**, 013612 (2014).

[273] B. Jackson, N. P. Proukakis, C. F. Barenghi, and E. Zaremba, *Finite-temperature vortex dynamics in Bose-Einstein condensates*, Phys. Rev. A **79**, 053615 (2009).

[274] J. R. Schloss and L. J. O'Riordan, *GPUE: Graphics Processing Unit Gross–Pitaevskii Equation solver*, J. Open Source Softw. **3**, 1037 (2018).

[275] S. Guo, S. M. Koh, A. R. Fritsch, I. B. Spielman, and J. P. Zwolak, *Combining machine learning with physics: A framework for tracking and sorting multiple dark solitons*, Phys. Rev. Research **4**, 023163 (2022).

[276] M. Kim, T. Rabga, Y. Lee, J. Goo, D. Bae, and Y.-i. Shin, *Suppression of Spontaneous Defect Formation in Inhomogeneous Bose Gases*, arXiv e-prints , arXiv:2208.02395 (2022).

[277] K. Kottmann, *Investigating Quantum Many-Body Systems with Tensor Networks, Machine Learning and Quantum Computers*, arXiv e-prints , arXiv:2210.11130 (2022).

[278] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor*, Nature Communications **5**, 1–7 (2014).

[279] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm*, arXiv e-prints , arXiv:1411.4028 (2014).

[280] Y. Li and S. C. Benjamin, *Efficient Variational Quantum Simulator Incorporating Active Error Minimization*, Phys. Rev. X **7**, 021050 (2017).

[281] S. Barison, F. Vicentini, and G. Carleo, *An efficient quantum algorithm for the time evolution of parameterized circuits*, Quantum **5**, 512 (2021).

[282] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, *An adaptive variational algorithm for exact molecular simulations on a quantum computer*, Nature Communications **10**, 3007 (2019).

[283] H. Abraham, AduOffei, R. Agarwal, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, E. Arbel, Arijit02, A. Asfaw, A. Avkhadiev, C. Azaustre, AzizNgoueya, A. Banerjee, A. Bansal, et al. *Qiskit: An Open-source Framework for Quantum Computing*, (2019).

[284] J. Spall, *An Overview of the Simultaneous Perturbation Method for Efficient Optimization*, Johns Hopkins Apl Technical Digest **19**, 482–492 (1998).

[285] A. Cervera-Lierta, *Exact Ising model simulation on a quantum computer*, Quantum **2**, 114 (2018).

[286] E. Bernstein and U. Vazirani, *Quantum Complexity Theory*, SIAM Journal on Computing **26**, 1411–1473 (1997).

[287] F. G. Brandão, W. Chemissany, N. Hunter-Jones, R. Kueng, and J. Preskill, *Models of Quantum Complexity Growth*, PRX Quantum **2**, 030316 (2021).

[288] H. T. Wang, B. Li, and S. Y. Cho, *Topological quantum phase transition in bond-alternating spin-$\frac{1}{2}$ Heisenberg chains*, Phys. Rev. B **87**, 054402 (2013).

[289] F. Pollmann, A. M. Turner, E. Berg, and M. Oshikawa, *Entanglement spectrum of a topological phase in one dimension*, Phys. Rev. B **81**, 064439 (2010).

[290] P. Sen, *Quantum phase transitions in the Ising model in a spatially modulated field*, Phys. Rev. E **63**, 016112 (2000).

[291] S. Bravyi, S. Sheldon, A. Kandala, D. C. McKay, and J. M. Gambetta, *Mitigating measurement errors in multiqubit experiments*, Physical Review A **103**, 042605 (2021).

[292] K. Kottmann, F. Metz, J. Fraxanet, and N. Baldelli. *Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer.* https://github.com/Qottmann/Quantum-anomaly-detection, (2021).

[293] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, *Discovering Physical Concepts with Neural Networks*, Phys. Rev. Lett. **124**, 010508 (2020).

[294] T. Szołdra, P. Sierant, M. Lewenstein, and J. Zakrzewski, *Unsupervised detection of decoupled subspaces: Many-body scars and beyond*, Phys. Rev. B **105**, 224205 (2022).

[295] S. Monaco, O. Kiss, A. Mandarino, S. Vallecorsa, and M. Grossi, *Quantum phase detection generalisation from marginal quantum neural network models*, arXiv e-prints , arXiv:2208.08748 (2022).

[296] J. Herrmann, S. Llima, A. Remm, P. Zapletal, N. McMahon, C. Scarato, F. Swiadek, C. Andersen, C. Hellings, S. Krinner, N. Lacroix, S. Lazar, M. Kerschbaum, D. Zanuz, G. Norris, M. Hartmann, A. Wallraff, and C. Eichler, *Realizing quantum convolutional neural networks on a superconducting quantum processor to recognize quantum phases*, Nature Communications **13**, 4144 (2022).

[297] Z. Chai, Y. Liu, M. Wang, Y. Guo, F. Shi, Z. Li, Y. Wang, and J. Du, *Quantum anomaly detection of audio samples with a spin processor in diamond*, arXiv e-prints , arXiv:2201.10263 (2022).

[298] G. Park, J. Huh, and D. K. Park, *Variational quantum one-class classifier*, arXiv e-prints , arXiv:2210.02674 (2022).

[299] V. S. Ngairangbam, M. Spannowsky, and M. Takeuchi, *Anomaly detection in high-energy physics using a quantum autoencoder*, Phys. Rev. D **105**, 095004 (2022).

[300] M. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. Coles, *Generalization in quantum machine learning from few training data*, Nature Communications **13**, 4919 (2022).

[301] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*, Phys. Rev. Lett. **70**, 1895–1899 (1993).

[302] A. K. Ekert, *Quantum cryptography based on Bell's theorem*, Phys. Rev. Lett. **67**, 661–663 (1991).

[303] C. H. Bennett and S. J. Wiesner, *Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states*, Phys. Rev. Lett. **69**, 2881–2884 (1992).

[304] V. Giovannetti, S. Lloyd, and L. Maccone, *Quantum Metrology*, Phys. Rev. Lett. **96**, 010401 (2006).

[305] M. Krenn, M. Malik, M. Erhard, and A. Zeilinger, *Orbital angular momentum of photons and the entanglement of Laguerre–Gaussian modes*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **375**, 20150442 (2017).

[306] M. A. Can, A. Klyachko, and A. Shumovsky, *Single-particle entanglement*, Journal of Optics B: Quantum and Semiclassical Optics **7**, L1–L3 (2005).

[307] Y. Li, M. Gessner, W. Li, and A. Smerzi, *Hyper- and hybrid nonlocality*, Phys. Rev. Lett. **120**, 050404 (2018).

[308] Y. Hasegawa, R. Loidl, G. Badurek, S. Filipp, J. Klepp, and H. Rauch, *Evidence for entanglement and full tomographic analysis of Bell states in a single-neutron system*, Phys. Rev. A **76**, 052108 (2007).

[309] M. Erhard, R. Fickler, M. Krenn, and A. Zeilinger, *Twisted Photons: New Quantum Perspectives in High Dimensions*, Light Sci Appl. **7**, 17146 (2017).

[310] L. Neves, G. Lima, A. Delgado, and C. Saavedra, *Hybrid photonic entanglement: Realization, characterization, and applications*, Phys. Rev. A **80**, 042322 (2009).

[311] Y. Aharonov, L. Davidovich, and N. Zagury, *Quantum random walks*, Phys. Rev. A **48**, 1687–1690 (1993).

[312] J. Kempe, *Quantum random walks: An introductory overview*, Contemp. Phys. **44**, 307–327 (2003).

[313] S. E. Venegas-Andraca, *Quantum walks: a comprehensive review*, Quantum Inf. Process. **11**, 1015–1106 (2012).

[314] R. Portugal, *Quantum Walks and Search Algorithms*, Springer New York (2013).

[315] E. Farhi and S. Gutmann, *Quantum computation and decision trees*, Phys. Rev. A **58**, 915–928 (1998).

[316] M. Bednarska, A. Grudka, P. Kurzyński, T. Łuczak, and A. Wójcik, *Quantum walks on cycles*, Physics Letters A **317**, 21–25 (2003).

[317] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders, *Quantum walks in higher dimensions*, J. Phys. A: Math. Gen. **35**, 2745–2753 (2002).

[318] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. *Quantum Walks on Graphs*. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, page 50–59, (2001).

[319] P. M. Preiss, R. Ma, M. E. Tai, A. Lukin, M. Rispoli, P. Zupancic, Y. Lahini, R. Islam, and M. Greiner, *Strongly correlated quantum walks in optical lattices*, Science **347**, 1229–1233 (2015).

[320] S. Mugel, A. Celi, P. Massignan, J. K. Asbóth, M. Lewenstein, and C. Lobo, *Topological bound states of a quantum walk with cold atoms*, Phys. Rev. A **94**, 023631 (2016).

[321] M. Karski, L. Förster, J.-M. Choi, A. Steffen, W. Alt, D. Meschede, and A. Widera, *Quantum Walk in Position Space with Single Optically Trapped Atoms*, Science **325**, 174–177 (2009).

[322] C. Robens, S. Brakhane, D. Meschede, and A. Alberti. *Quantum Walks with Neutral Atoms: Quantum Interference Effects of One and Two Particles*, pages 1–15. (2016).

[323] J.-Q. Zhou, L. Cai, Q.-P. Su, and C.-P. Yang, *Protocol of a quantum walk in circuit QED*, Phys. Rev. A **100**, 012343 (2019).

[324] E. Flurin, V. V. Ramasesh, S. Hacohen-Gourgy, L. S. Martin, N. Y. Yao, and I. Siddiqi, *Observing Topological Invariants Using Quantum Walks in Superconducting Circuits*, Phys. Rev. X **7**, 031023 (2017).

[325] H. Schmitz, R. Matjeschk, C. Schneider, J. Glueckert, M. Enderlein, T. Huber, and T. Schaetz, *Quantum Walk of a Trapped Ion in Phase Space*, Phys. Rev. Lett. **103**, 090504 (2009).

[326] F. Zähringer, G. Kirchmair, R. Gerritsma, E. Solano, R. Blatt, and C. F. Roos, *Realization of a Quantum Walk with One and Two Trapped Ions*, Phys. Rev. Lett. **104**, 100503 (2010).

[327] L. Neves and G. Puentes, *Photonic Discrete-time Quantum Walks and Applications*, Entropy **20**, 731 (2018).

[328] T. Giordani, E. Polino, S. Emiliani, A. Suprano, L. Innocenti, H. Majury, L. Marrucci, M. Paternostro, A. Ferraro, N. Spagnolo, and F. Sciarrino, *Experimental Engineering of Arbitrary Qudit States with Discrete-Time Quantum Walks*, Phys. Rev. Lett. **122**, 020503 (2019).

[329] A. Crespi, R. Osellame, R. Ramponi, V. Giovannetti, R. Fazio, L. Sansoni, F. De Nicola, F. Sciarrino, and P. Mataloni, *Anderson localization of entangled photons in an integrated quantum walk*, Nat. Photonics **7**, 322–328 (2013).

[330] C. A. Ryan, M. Laforest, J. C. Boileau, and R. Laflamme, *Experimental implementation of a discrete-time quantum random walk on an NMR quantum-information processor*, Phys. Rev. A **72**, 062317 (2005).

[331] J. Du, H. Li, X. Xu, M. Shi, J. Wu, X. Zhou, and R. Han, *Experimental implementation of the quantum random-walk algorithm*, Phys. Rev. A **67**, 042316 (2003).

[332] K. Kadian, S. Garhwal, and A. Kumar, *Quantum walk and its application domains: A systematic review*, Computer Science Review **41**, 100419 (2021).

[333] N. Shenvi, J. Kempe, and K. B. Whaley, *Quantum random-walk search algorithm*, Phys. Rev. A **67**, 052307 (2003).

[334] A. Ambainis, J. Kempe, and A. Rivosh. *Coins Make Quantum Walks Faster.* In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 1099–1108, USA, (2005). Society for Industrial and Applied Mathematics.

[335] F. Caruso, *Universally optimal noisy quantum walks on complex networks*, New Journal of Physics **16**, 055015 (2014).

[336] F. Caruso, A. Crespi, A. Ciriolo, F. Sciarrino, and R. Osellame, *Fast escape of a quantum walker from an integrated photonic maze*, Nature Communications **7**, 11682 (2016).

[337] S. Dernbach, A. Mohseni-Kabir, S. Pal, D. Towsley, and M. Gepner, *Quantum Walk Inspired Neural Networks for Graph-Structured Data*, arXiv e-prints , arXiv:1801.05417 (2018).

[338] L. S. de Souza, J. H. de Carvalho, and T. A. Ferreira. *Quantum Walk to Train a Classical Artificial Neural Network.* In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS).* IEEE, (2019).

[339] M. Schuld, I. Sinayskiy, and F. Petruccione, *Quantum walks on graphs representing the firing patterns of a quantum neural network*, Phys. Rev. A **89**, 032333 (2014).

[340] R. Vieira, E. P. M. Amorim, and G. Rigolin, *Dynamically Disordered Quantum Walk as a Maximal Entanglement Generator*, Phys. Rev. Lett. **111**, 180503 (2013).

[341] R. Vieira, E. P. M. Amorim, and G. Rigolin, *Entangling power of disordered quantum walks*, Phys. Rev. A **89**, 042307 (2014).

[342] M. Zeng and E. H. Yong, *Discrete-Time Quantum Walk with Phase Disorder: Localization and Entanglement Entropy*, Sci. Rep. **7**, 12024 (2017).

[343] C. M. Chandrashekar, *Disorder induced localization and enhancement of entanglement in one- and two-dimensional quantum walks*, arXiv e-prints , arXiv:1212.5984 (2012).

[344] L. Innocenti, H. Majury, T. Giordani, N. Spagnolo, F. Sciarrino, M. Paternostro, and A. Ferraro, *Quantum state engineering using one-dimensional discrete-time quantum walks*, Phys. Rev. A **96**, 062326 (2017).

[345] A. Gratsea, M. Lewenstein, and A. Dauphin, *Generation of hybrid maximally entangled states in a one-dimensional quantum walk*, Quantum Sci. Technol. **5**, 025002 (2020).

[346] I. Carneiro, M. Loo, X. Xu, M. Girerd, V. Kendon, and P. L. Knight, *Entanglement in coined quantum walks on regular graphs*, New J. Phys. **7**, 156–156 (2005).

[347] S. Salimi and R. Yosefjani, *Asymptotic entanglement in 1D quantum walks with a time-dependent coined*, Int. J. Mod. Phys. B **26**, 1250112 (2012).

[348] N. P. Kumar, R. Balu, R. Laflamme, and C. M. Chandrashekar, *Bounds on the dynamics of periodic quantum walks and emergence of the gapless and gapped Dirac equation*, Phys. Rev. A **97**, 012116 (2018).

[349] N. Lo Gullo, C. V. Ambarish, T. Busch, L. Dell'Anna, and C. M. Chandrashekar, *Dynamics and energy spectra of aperiodic discrete-time quantum walks*, Phys. Rev. E **96**, 012111 (2017).

[350] P. Ribeiro, P. Milman, and R. Mosseri, *Aperiodic Quantum Random Walks*, Phys. Rev. Lett. **93**, 190503 (2004).

[351] C. Watkins and P. Dayan, *Technical Note: Q-Learning*, Machine Learning **8**, 279–292 (1992).

[352] R. Reuvers, *An algorithm to explore entanglement in small systems*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**, 20180023 (2018).

[353] A. C. Orthey and E. P. M. Amorim, *Connecting velocity and entanglement in quantum walks*, Phys. Rev. A **99**, 032320 (2019).

[354] C. M. Chandrashekar, R. Srikanth, and S. Banerjee, *Symmetries and noise in quantum walk*, Phys. Rev. A **76**, 022316 (2007).

[355] M. Hinarejos, C. Di Franco, A. Romanelli, and A. Pérez, *Chirality asymptotic behavior and non-Markovianity in quantum walks on a line*, Phys. Rev. A **89**, 052330 (2014).

[356] T. A. Brun, H. A. Carteret, and A. Ambainis, *Quantum random walks with decoherent coins*, Phys. Rev. A **67**, 032304 (2003).

[357] S.-J. Tao, Q.-Q. Wang, Z. Chen, W.-W. Pan, S. Yu, G. Chen, X.-Y. Xu, Y.-J. Han, C.-F. Li, and G.-C. Guo, *Experimental optimal generation of hybrid entangled states in photonic quantum walks*, Opt. Lett. **46**, 1868–1871 (2021).

[358] R. Zhang, R. Yang, J. Guo, C.-W. Sun, J.-C. Duan, H. Zhou, Z. Xie, P. Xu, Y.-X. Gong, and S.-N. Zhu, *Maximal coin-walker entanglement in a ballistic quantum walk*, Phys. Rev. A **105**, 042216 (2022).

[359] C. B. Naves, M. A. Pires, D. O. Soares-Pinto, and S. M. D. Queirós, *Enhancing entanglement with the generalized elephant quantum walk from localized and delocalized states*, Phys. Rev. A **106**, 042408 (2022).

[360] P. Rajauria, P. Chawla, and C. M. Chandrashekar, *Estimation of one-dimensional discrete-time quantum walk parameters by using machine learning algorithms*, arXiv e-prints , arXiv:2007.04572 (2020).

[361] M. M. Rams, P. Sierant, O. Dutta, P. Horodecki, and J. Zakrzewski, *At the Limits of Criticality-Based Quantum Metrology: Apparent Super-Heisenberg Scaling Revisited*, Phys. Rev. X **8**, 021022 (2018).

[362] S. Pang and A. N. Jordan, *Optimal adaptive control for quantum metrology with time-dependent Hamiltonians*, Nature Communications **8**, 14695 (2017).

[363] G. Matos, S. Johri, and Z. Papić, *Quantifying the Efficiency of State Preparation via Quantum Variational Eigensolvers*, PRX Quantum **2**, 010309 (2021).

[364] A. G. R. Day, M. Bukov, P. Weinberg, P. Mehta, and D. Sels, *Glassy Phase of Optimal Quantum Control*, Phys. Rev. Lett. **122**, 020601 (2019).

[365] E. Farhi and A. W. Harrow, *Quantum Supremacy through the Quantum Approximate Optimization Algorithm*, arXiv e-prints , arXiv:1602.07674 (2016).

[366] P. Doria, T. Calarco, and S. Montangero, *Optimal Control Technique for Many-Body Quantum Dynamics*, Phys. Rev. Lett. **106**, 190501 (2011).

[367] S. van Frank, M. Bonneau, J. Schmiedmayer, S. Hild, C. Gross, M. Cheneau, I. Bloch, T. Pichler, A. Negretti, T. Calarco, and et al., *Optimal control of complex atomic quantum systems*, Scientific Reports **6**, 34187 (2016).

[368] J. H. M. Jensen, F. S. Møller, J. J. Sørensen, and J. F. Sherson, *Achieving fast high-fidelity optimal control of many-body quantum dynamics*, Phys. Rev. A **104**, 052210 (2021).

[369] O. Lockwood and M. Si, *Reinforcement Learning with Quantum Variational Circuits*, arXiv e-prints , arXiv:2008.07524 (2020).

[370] V. Dunjko, J. M. Taylor, and H. J. Briegel. *Advances in quantum reinforcement learning.* In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287, (2017).

[371] S. Jerbi, L. M. Trenkwalder, H. Poulsen Nautrup, H. J. Briegel, and V. Dunjko, *Quantum Enhancements for Deep Reinforcement Learning in Large Spaces*, PRX Quantum **2**, 010328 (2021).

[372] V. Saggio, B. E. Asenbeck, A. Hamann, T. Strömberg, P. Schiansky, V. Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund, and et al., *Experimental quantum speed-up in reinforcement learning agents*, Nature **591**, 229–233 (2021).

[373] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, et al., *Quantum phases of matter on a 256-atom programmable quantum simulator*, Nature **595**, 227–232 (2021).

[374] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, *Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms*, Journal of Magnetic Resonance **172**, 296–305 (2005).

[375] H. van Hasselt, A. Guez, and D. Silver, *Deep Reinforcement Learning with Double Q-learning*, arXiv e-prints , arXiv:1509.06461 (2015).

[376] D. Guéry-Odelin, A. Ruschhaupt, A. Kiely, E. Torrontegui, S. Martínez-Garaot, and J. G. Muga, *Shortcuts to adiabaticity: Concepts, methods, and applications*, Rev. Mod. Phys. **91**, 045001 (2019).

[377] H. Lamm and S. Lawrence, *Simulation of Nonequilibrium Dynamics on a Quantum Computer*, Phys. Rev. Lett. **121**, 170501 (2018).

[378] D. Poulin, A. Qarry, R. Somma, and F. Verstraete, *Quantum Simulation of Time-Dependent Hamiltonians and the Convenient Illusion of Hilbert Space*, Phys. Rev. Lett. **106**, 170501 (2011).

[379] W. H. Zurek, U. Dorner, and P. Zoller, *Dynamics of a Quantum Phase Transition*, Phys. Rev. Lett. **95**, 105701 (2005).

[380] F. Barratt, J. Dborin, M. Bal, V. Stojevic, F. Pollmann, and A. G. Green, *Parallel quantum simulation of large systems on small NISQ computers*, npj Quantum Information **7**, 79 (2021).

[381] S.-H. Lin, R. Dilip, A. G. Green, A. Smith, and F. Pollmann, *Real- and Imaginary-Time Evolution with Compressed Quantum Circuits*, PRX Quantum **2**, 010342 (2021).

[382] S.-J. Ran, *Encoding of matrix product states into quantum circuits of one- and two-qubit gates*, Phys. Rev. A **101**, 032310 (2020).

[383] M. S. Rudolph, J. Chen, J. Miller, A. Acharya, and A. Perdomo-Ortiz, *Decomposition of Matrix Product States into Shallow Quantum Circuits*, arXiv e-prints , arXiv:2209.00595 (2022).

[384] M. Ben Dov, D. Shnaiderov, A. Makmal, and E. G. Dalla Torre, *Approximate encoding of quantum states using shallow circuits*, arXiv e-prints , arXiv:2207.00028 (2022).

[385] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, *Towards quantum machine learning with tensor networks*, Quantum Science and Technology **4**, 024001 (2019).

[386] J. Dborin, F. Barratt, V. Wimalaweera, L. Wright, and A. G. Green, *Matrix product state pre-training for quantum machine learning*, Quantum Science and Technology **7**, 035014 (2022).

[387] M. L. Wall, M. R. Abernathy, and G. Quiroz, *Generative machine learning with tensor networks: Benchmarks on near-term quantum computers*, Phys. Rev. Research **3**, 023010 (2021).

[388] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, *Quantum Fingerprinting*, Phys. Rev. Lett. **87**, 167902 (2001).

[389] D. Gottesman and I. Chuang, *Quantum Digital Signatures*, arXiv e-prints , quant–ph/0105032 (2001).

[390] See ancillary files on arXiv for three movies displaying Bloch sphere trajectories of spins controlled by QMPS agents.

[391] F. Metz and M. Bukov. *Self-Correcting Quantum Many-Body Control using Reinforcement Learning with Tensor Networks.* `https://github.com/frmetz/QMPS`, (2022).

[392] S. Lu, M. Kanász-Nagy, I. Kukuljan, and J. I. Cirac, *Tensor networks and efficient descriptions of classical data*, arXiv e-prints , arXiv:2103.06872 (2021).

[393] T. Schaul, D. Horgan, K. Gregor, and D. Silver. *Universal Value Function Approximators.* In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, (2015). PMLR.