ICFO
THE INSTITUTE OF PHOTONIC SCIENCES

DOCTORAL THESIS

A machine learning ride in the physics theme park: from quantum to biophysics

Author:
Borja REQUENA

Supervisors:
Prof. Dr. Maciej LEWENSTEIN
Dr. Gorka MUÑOZ-GIL

March 2024

Abstract

The integration of artificial intelligence into research is propelling progress and discoveries across the entire scientific landscape. Artificial intelligence tools boost the development of novel scientific insights and theories by processing extensive data sets, guiding exploration and hypothesis formation, enhancing experimental setups, and even enabling autonomous discovery. In this thesis, we harness the power of machine learning, a sub-field of artificial intelligence, to study non-deterministic systems, which are amongst the hardest to characterize.

On one hand, we address problems inherent to the study of quantum systems and the development of quantum technologies. Quantum physics presents formidable challenges due to the associated exponential complexity with the size of the system at hand, as well as its intrinsic stochastic nature and the presence of intricate correlations between its components. We employ reinforcement learning, a machine learning technique that excels at dealing with vast hypothesis spaces, to address some of these challenges. Notably, reinforcement learning has demonstrated super-human performance in multiple complex games like Go, which present similar characteristics to the problems encountered in the study of quantum physics. We use it to systematically simplify complex common problems in condensed matter and quantum information processing tasks, as well as to implement robust calibration schemes for quantum computers.

On the other hand, we focus on the characterization of complex stochastic processes, such as diffusion. Understanding diffusion processes is crucial to unravel the complex underlying physical and biological mechanisms governing them. This involves extracting meaningful parameters from the analysis of stochastic trajectories described by tracked particles. However, accurately capturing and analyzing the trajectories presents multiple challenges, stemming from the combination of their random nature, complex dynamics, and experimental drawbacks, such as noise. We develop machine learning algorithms to accurately extract such parameters, even when they vary with time, and demonstrate their applicability in experimental scenarios. Furthermore, we apply similar techniques to study the diffusion of internet users browsing an e-commerce website, predicting their likelihood to make a purchase before closing the session.

Resum

La integració de la intel·ligència artificial a la recerca accelera el progrés cap a nous descobriments en tot l'àmbit científic. Les eines d'intel·ligència artificial contribueixen al desenvolupament de noves teories i coneixements processant grans quantitats de dades, guiant l'exploració i la formulació d'hipòtesis, millorant els experiments i, fins i tot, fent possible descobriments automàtics. En aquesta tesi, aprofitem el poder de l'aprenentatge automàtic ("machine learning"), un camp de la intel·ligència artificial, per estudiar sistemes no-deterministes, que es troben entre els més difícils de caracteritzar.

Per una banda, tractem problemes inherents a l'estudi de sistemes quàntics i del desenvolupament de noves tecnologies quàntiques. La física quàntica planteja reptes formidables derivats de la complexitat exponencial amb la mida del sistema considerat, en combinació amb una naturalesa intrínsicament estocàstica i la presència de correlacions complexes entre elements del sistema. Per tractar alguns d'aquests reptes, fem servir aprenentatge de reforç ("reinforcement leraning"), una tècnica de l'aprenentatge automàtic capaç d'explorar grans espais d'hipòtesis. Per exemple, empleant aquestes tècniques, s'ha aconseguit superar als millors jugadors del món en jocs complexes com el Go, que presenten característiques similars a problemes emergents en l'estudi de la quàntica. En el nostre cas, desenvolupem mètodes per simplificar problemes complexes comuns en els camps de la matèria condensada i de la informació quàntica de forma sistemàtica, i disenyem protocols robustos de cal·libració d'ordinadors quàntics.

Per l'altra banda, ens dediquem a la caracterització de processos estocàstics complexes, com és la difusió. Entendre els processos de difusió és essencial per descobrir els mecanismes físics i biològics que els governen, el que comporta l'anàlisi de trajectòries estocàstiques descrites per partícules per tal d'extruere'n paràmetres significatius del sistema. Aquest anàlisi, però, presenta grans reptes des de l'adquisició de les trajectòries fins al seu estudi posterior que provenen, principalment, de la combinació de la seva naturales aleatòria, amb la presència de dinàmiques complexes i altres inconvenients experimentals, com ara el soroll. Utilitzant tècniques d'aprenentatge automàtic, desenvolupem algoritmes per analitzar aquestes trajectòries i extreure'n els paràmetres d'interès acuradament, fins i tot quan aquests canvien amb el temps. Després, utilitzem aquests algoritmes per estudiar diferents experiments en sistemes biològics i, també, per estudiar les trajectòries descrites per usuaris navegant una pàgina de comerç online. En aquest últim cas, en comptes d'extreure paràmetres físics, inferim si l'usuari farà una compra abans de tancar la sessió.

Acknowledgements

This is a page of gratitude to everyone who has partaken in the wild ride this thesis has been. This adventure could have never reached such peak levels of awesomeness without the wonderful people who have tagged along and the love we share. Little did I know when I started I could now call so many people dear friends, and while I'm happy to reach the end, the journey is the real treasure I cherish!

Of course, this wouldn't have been possible without the one and only Maciej Lewenstein. I couldn't be more grateful to work in this special environment you nourish, and I wish every PhD student could have a QOT-like experience. Furthermore, I could never forget the support from Ferran Mazzanti and Lucas Lacasa, who gave everything they could to help me forge my path forward.

Gorka, I feel extremely fortunate to work with you. Thank you for sharing an enthusiastic approach to life, and teaching me so much, from physics and machine learning to pushing my boundaries in the wall and killing it in deep powder. Thank you for taking me on wild adventures (your daily life), and teaching me that science is best done with friends. It has been an absolute pleasure!

Joana and Niccolò, you are the best partners in crime I could have asked for. I have enjoyed every single moment with you since our times in the *purgatory*, and you have made this journey an absolute blast. Thank you for the many Solsones, Ivalos (♡), barbeques, parties, torture-like climbing sessions, and many more! Thank you for everything, I love you!!

Alex, you are a blessing to this world! Thank you for the intense wild discussions, the huge ice-creams in the beach, and the much needed support to take good care of Pipo! (we do not talk about the ABM)

In QOT, I've found plenty of terrific people. Quel c***o di Paolo, che amo profondamente. Gabriel, thanks for loving cool things! Ania, you're simply amazing!! My new office mates that regularly suffer my terrible Japanese (sorry Yuma!). The original gangsters Albert, Dani, Sergi, Jessica, Christos, Tymek, Emilio, Tobi, Valentin, ... And all the new buddies. Thank you all for a lovely daily life!

At ICFO, I've met extremely brilliant people. Thanks to Jordi and Vedran for the weirdest project ever, and Carlo for making sense out of my messes. My deepest gratitude to the *futbolín* crew and the QML reading group, which have gifted me some of my fondest memories. Especially, thanks to Nico, Natàlia, Roberto, Juan, Fèlix, Korbinian, Alex P.-K., Patrick, and Joe for making these things possible. Huge thanks to the most catalan italian friends for the wildest parties and most random conversations, and to my fellow *scalata* mates for keeping me safe and motivated.

I am very thankful to the people at Xanadu that took me on board for an entire summer. Special thanks for Tom, Nathan and Korbinian, to the PennyLane core team, especially Matthew and Christina, and to Vincent for making PennyLane go brr together! Of course, this would have not been the same without the greatest fellow interns Frederik, Ludmilla, Oriel (les quatre heures!), Rio, Ryk, Noro, Jacob, Danial, and Edy. I love you guys!

En esta aventura, me acompañan personas que cimientan mi vida y nutren mi día a día. Por un lado, cuento con el apoyo incondicional de toda mi familia, que me impulsa a crecer a diario con una sonrisa. Per altra banda, estic envoltat de gent que fan la vida divertida. Moltíssimes gràcies a la meva companyia de software preferida *Arisoft*, a les habitants de *Villa Lola*, i als millors de tota la vida Lena, Berta, Hernán, Marcel, Judit i Dani. Finalment, amor infinit des del més profund del meu cor al Bernat i a l'Alba, que us tinc sempre tan a prop.

List of publications

Publications included in this thesis

1. **B. Requena**, G. Cassani, J. Tagliabue, C. Greco, L. Lacasa. *Shopper intent prediction from clickstream e-commerce data with minimal browsing information.* Scientific Reports, **10**, 16983 (2020).

2. G. Muñoz-Gil, *et. al.*, **B. Requena**, *et. al. Objective comparison of methods to decode anomalous diffusion.* Nature Communications, **12**, 6253 (2021).

3. **B. Requena**, G. Muñoz-Gil, M. Lewenstein, V. Dunjko, J. Tura. *Certificates of quantum many-body properties assisted by machine learning*. Physical Review Research, **5**, 013097 (2023).

4. **B. Requena**, S. Masó-Orriols, J. Bertran, M. Lewenstein, C. Manzo, G. Muñoz-Gil. *Inferring pointwise diffusion properties of single trajectories with deep learning.* Biophysical Journal, **122**(22), 4360 - 4369 (2023).

5. A. Dawid, J. Arnold*, **B. Requena***, A. Gresch*, *et. al. Machine learning in quantum sciences.* To be published with Cambridge University Press (July 2024). ArXiv:2204.04198. (* equal contribution)

Open-source libraries and educational material developped within this thesis

- **B. Requena**. *AnDi-Unicorns Python library.*
  GitHub BorjaRequena/AnDi-unicorns (2020).

- **B. Requena**. *Introduction to variational Monte Carlo with neural network quantum states*. brequena.com (2021).

- **B. Requena**, G. Muñoz-Gil, J. Tura. *BOUNCE Python library.*
  Zenodo DOI: 10.5281/zenodo.4585623 (2021).

- G. Muñoz-Gil, **B. Requena**, G. Volpe, M. A. Garcia-March, C. Manzo. *AnDi_datasets Python library.*
  Zenodo DOI: 10.5281/zenodo.4775310 (2021).

- **B. Requena**, G. Muñoz-Gil. *STEP Python library.*
  Zenodo DOI: 10.5281/zenodo.7480413 (2022).

- **B. Requena**, M. Płodzień, P. Stornati, A. Dauphin. *Neural network course*.
  GitHub BorjaRequena/Neural-Network-Course (2022).

- **B. Requena**. *GPT: the model even your dog has heard of*.
  GitHub BorjaRequena/GPTutorial (2023).

- **B. Requena**. *Compilation of quantum circuits*. PennyLane Demos (2023).

- **B. Requena**. *Gate calibration with reinforcement learning*. PennyLane Demos (2024).

Contents

A mis mayores héroes: Ángel, Francisca, Pedro y Dolores.

Chapter 1

Introduction

1.1 The scope of this thesis

In this thesis, we focus on currently relevant open research questions and areas of scientific interest, and identify consistent nuances and limitations therein that hinder the scientific progress. The main objective is to develop the necessary tools, in the form of novel open-source algorithms and methodologies, to overcome these challenges. Our goal is that these technological advancements serve as catalysts to push forward the respective areas of research. For this reason, we take special care to ensure accessibility and a broad impact, carefully packaging our contributions with abundant teaching material.

With these principles, we not only seek a deeper understanding of nature, but we also wish to actively contribute to the development of the scientific ecosystem through tangible technological advancements. Hence, we frame the entire thesis from this perspective, placing the emphasis on the influential role that emerging technologies, particularly artificial intelligence (AI), play in science. We illustrate how we harness the power of machine learning (ML), a sub-field of AI, to overcome the challenges inherent in in our chosen fields of study: the development of quantum technologies, and the study of diffusive processes.

1.2 Scientific research and technological development

Why focus on technological development? In our view, scientific and technological development exist in a dynamic and symbiotic relationship, each propelling the other to new heights of discovery and innovation. In scientific research, the formulation of questions and the pursuit of knowledge are intrinsically related to the available tools and resources. The continuous technological progress provides researchers with increasingly sophisticated instruments and methodologies, expanding the scope of what can be explored. Simultaneously, as science advances, unveiling new phenomena and raising new questions, it motivates the development of innovative solutions. Furthermore, the advances of the boundaries of knowledge fuel the evolution of new technologies. This reciprocal dynamic is particularly evident at the forefront of research, where cutting-edge investigations require the creation of novel tools and methodologies.

Throughout history, we encounter multiple prominent examples of such interconnection that have shaped the landscape of modern science, medicine, and our very daily lives. Notably, the mid-20th century witnessed remarkable strides in the fields of optics and photonics culminating in the birth of the laser [1]. Stemming from Albert Einstein's work on stimulated emission of radiation [2], the laser stands

as one of the most transformative technologies of the past century. Beyond its economic impact as a multi-billion dollar industry due to its countless applications, the laser has been instrumental in groundbreaking scientific discoveries, such as the detection of gravitational waves [3]. Additionally, it has paved the way for the exploration of entirely new research avenues, for instance, enabling research in attosecond physics [4]. Both examples recently laureated by the Nobel prize in physics in 2017 [5] and 2023 [6], respectively. Lasers also assume a pivotal role in the development of quantum computers and simulators, as well as in advanced microscopy techniques, which are of particular relevance for this thesis. Now, a compelling question arises: what novel technological developments will arise from the insights gained in these fields?

This interplay between scientific progress and technological innovation echoes the development of the transistor [7], the cornerstone of modern electronics. Resulting from the exploration of semiconductors, and propelled by advances in quantum and solid-state physics, the transistor catalyzed the development of computational devices. Ever since their inception, computers have aided scientific research [8], enabling the simulation of intricate phenomena, modeling complex systems, and processing vast data sets with unprecedented efficiency. Furthermore, the ongoing refinement of semiconductor-based technology has not only fueled the exponential evolution of computers, but has also given raise to hardware accelerators, such as graphics processing units (GPUs). These advances have been essential for the development of the promising field of AI, which is reshaping the world and the scientific landscape.

This interdependence continues to drive advancements that redefine the possibilities within the scientific realm, as exemplified by the legacies of the laser and the transistor. In this thesis, draw from the transformative potential that emerges when scientific curiosity converges with technological ingenuity, designing AI-based tools to push the frontiers of research across several fields in physics.

1.3 Scientific discovery in the age of artificial intelligence

In the past decade, AI has emerged as a major transformative force in our society. Its influence extends across a wide range of domains, redefining how we work, communicate, and engage with the world. From smart navigation systems and personalized digital media content, to the advent of autonomous vehicles and virtual assistants, AI has become an integral part of our daily lives. Beyond our individual conveniences, AI is revolutionizing entire industrial sectors by streamlining automation processes and and offering data-driven insights. In medicine, it plays an ever-increasing role in diagnostics, drug discovery, and the creation of personalized treatment plans, collectively contributing to improved patient outcomes.

Science is perhaps the domain where AI holds the potential for the most significant impact.[1] Fundamentally, the development of novel scientific insights and theories is tied to the collection, processing and comprehension of data. AI plays a crucial role boosting this kind of scientific progress by processing extensive data sets, guiding exploration and hypothesis formation, and even providing tools for autonomous discovery [9].

These techniques transcend scientific disciplines. The impact of AI-driven simulations reverberates across the scientific landscape, from molecular dynamics in

[1] Or so I (Borja) like to believe.

chemistry [10] to climate modeling in environmental sciences [11]. AI-based methods to automatically collect, annotate, process and visualize massive data sets play an essential role in fields ranging from astrophysics [12] to biochemistry [13]. AI-based tools are indispensable to explore vast hypothesis spaces, such as the 10^{60} drug candidates [14] or the 10^{180} possible stable materials, to help us design new molecules [15], identify new particles [16], or formulate new mathematical theories [17, 18]. A monumental breakthrough in this context has been the development of *AlphaFold* for the protein-folding problem [19], providing researchers with a comprehensive data base containing hundreds of thousands of protein structures that includes the entire human proteome [20]. Furthermore, AI algorithms excel at drawing hypotheses from observations to control or even suggest new experiments, for example, to synthesize new compounds in automatic labs [21], prepare quantum states [22], or control fusion reactors [23].

The influence of AI extends to the point of creating new paradigms of scientific discovery. Examining the history of science, the first paradigm dates back thousands of years ago, and it is based on the direct empirical observation of the patterns of nature. The second paradigm is characterized by the theoretical description of nature derived from empirical observations. It was consolidated in the seventeenth century with the formulation of the laws of motion and gravitation by Sir Isaac Newton. Subsequent prominent examples include Maxwell's equations of electromagnetism in the nineteenth century, and Schrödinger's equation in the twentieth century. However, despite the precision with which these fundamental equations describe nature, they are only analytically solvable in very simple scenarios. It wasn't until digital computers emerged later on that century, that these equations could be solved more broadly, leading to the third paradigm of numerical computation.

Early in the twenty-first century, we have witnessed the emergence of the fourth paradigm: data-intensive scientific discovery [24]. This paradigm focuses on developing technologies to extract scientific insights from vast amounts of data, recognizing that data collection has far outpaced analysis capabilities. ML thrives in this data-intensive ecosystem. For instance, it serves as an indispensable tool processing hundreds of terabytes generated every second in particle collision experiments [25], monitoring ecological environments in real-time [26], or even parsing the huge corpus of scientific publications to identify new research avenues to explore [27] or evaluate scientific achievements [28].

Currently, we find ourselves at what could potentially be the dawn of a fifth paradigm [29]. As all the paradigms coexist and complement each other, fundamental equations continue to be solved at scale. These numerical solutions, akin to highly accurate simulations of the natural world, come at a high computational cost. However, their intermediate steps and details can be leveraged to train ML models that can perform new calculations orders of magnitude faster with some trade-offs in accuracy. This allows to, for example, study phenomena at longer time-scales [30], or it may even enable the creation of hierarchies of simulations to reach previously unimaginable scales. What possibilities will this technology unfold for us in the coming future?

AI stands as a vibrant field of research on its own. As it advances, novel techniques gradually permeate the scientific domain from other contexts. For example, the remarkable *AlphaZero* algorithm [31], which defeated the world champions in chess, go and shogi in 2017, lead to the discovery of new matrix multiplication and sorting algorithms half a decade later [32, 33]. Notable advances such as Google's *BERT* algorithm [34], initially developed to enhance their search engine, are currently used in protein design [35]. Moreover, similar principles derived from it have

contributed to a deeper understanding of viral mutations, particularly those leading to viral escape [36].

In conclusion, AI plays an increasingly important role in scientific discovery, accelerating, enhancing and enabling research across disciplines. As it continues to evolve, the synergy between AI and scientific research is poised to drive breakthroughs, shape new paradigms, and propel humanity to an era of unprecedented knowledge and discovery.

1.4 The structure of this thesis

This thesis is organized in an introductory chapter followed by two main thematic parts, each encompassing several chapters. In the introductory Chapter 2, we review the main concepts of ML that lay the foundations to understand the subsequent parts of the thesis. Starting from the fundamentals, we dive into advanced techniques and algorithms.

In the first part of the thesis, we explore the application of ML algorithms to the study of quantum physics and the development of quantum technologies. Starting with Chapter 3, we provide an overview of the primary challenges inherent to these fields and review notable applications of ML therein. Then, we proceed to introduce our contributions to the field, both based on reinforcement learning (RL). In Chapter 4, we present a method to systematically simplify and approximate complex paradigmatic problems in the study of quantum many-body systems and quantum information processing. In Chapter 5, we implement a scheme to design quantum gates for superconducting quantum computers that are robust to noise.

The second part of the thesis is dedicated to the application of ML algorithms in the study diffusion processes. In Chapter 6, we introduce the foundations of diffusion and anomalous diffusion, as well as the role that ML algorithms play in the characterization of diffusive trajectories. In Chapters 7 and 8, we present novel ML methods developed to study diffusion trajectories with constant and time-dependent properties, respectively. In Chapter 9, we investigate the diffusion patterns of internet users browsing an e-commerce website to predict whether they will purchase an item before closing the session.

Finally, we conclude with Chapter 10, where we provide an overview of the thesis and the conclusions extracted from each of the main parts.

Chapter 2

Preliminaries

In this chapter, we review the main concepts of machine learning (ML) that are necessary to understand the upcoming chapters of this thesis. We start by introducing the fundamentals, followed by a few examples of typical learning schemes, to show what "learning" actually means. We complement these basic concepts with more advanced ideas of great practical importance, such as what overfitting is and how to avoid it. We proceed with an introduction to deep learning (DL), motivating its use, and describing some of the most prominent architectures involved in the following chapters. Finally, we finish with an introduction to reinforcement learning (RL), featuring two prototypical algorithms that are used in Chapters 4 and 5.

Most of the content of this chapter is based on our book on ML for the quantum sciences, and our practical hands-on ML course, both respectively accessible in Refs. [37] and [38].

2.1 Fundamentals of machine learning

As science and technology advance, we encounter increasingly complex or abstract problems that are hard to formalize mathematically. For instance, the problem of face recognition in images cannot be easily presented in a formal mathematical way, or tasks such as detecting new phases of matter may not even have a known mathematical formulation. Hence, these types of problems cannot be effectively addressed using standard hard-coded algorithms.

The field of ML emerges as a new paradigm to develop algorithms that are not explicitly programmed, but learned from experience instead, typically in the form of data. Fig. 2.1 provides a visual comparison between traditional programming and ML. This process heavily relies on applied statistics, emphasizing the use of computers to approximate complex functions. This trend continues with the rise of DL [39], which employs parametrized hierarchical models to extract intricate patterns from data, achieving remarkable accuracy across various tasks.

In short, an ML algorithm is an algorithm capable of learning from data. We can consider a computer program to learn from experience with respect to some class of tasks and performance measure if its performance in those tasks improves with experience [40]. Hence, there are a few fundamental ingredients to any learning algorithm:

A task to solve: Tasks are often defined in terms of how examples or data instances should be processed. These are usually represented by vectors $\boldsymbol{x} \in \mathbb{R}^n$ where every entry is a *feature*. We show some typical ML tasks in Section 2.1.1.

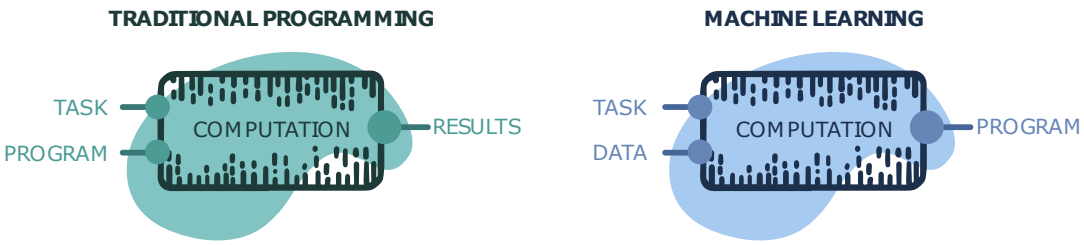

FIGURE 2.1: **Comparison between traditional programming and ML.** In traditional programming, an input task and a program to solve it yield the desired result after some computation. In the ML paradigm, an input task and data instances result in a program to solve the task. The data can be, for instance, pairs of problem examples and their solutions.

A performance measure: A consistent quantitative measure of the ML algorithm's performance on the task. These metrics can range from simple direct comparisons between the obtained and expected output, to more involved and task-specific functions.

Data (experience): A data set containing a collection of data instances, potentially including the desired output $\boldsymbol{y}$ for each of them $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$. We consider different types of learning depending on the kind of data set the algorithm is allowed to experience, as detailed in Section 2.1.2.

A model: The structure that encodes the resulting program. This can range from a parametrized linear function, to a combination of deep neural networks (NNs).

In these terms, the learning process can be described as the iterative maximization of the model's performance on the given task and data.

2.1.1 Typical machine learning tasks

The main ingredient in the learning process is a task. It is important to distinguish the objective task with the learning process itself. The latter is the process through which the ability to develop the task is acquired. For example, if we would like a robot to walk, walking is the task [39].

There are very diverse tasks that can be solved with ML. Here, we describe some of the archetypical ML tasks that are related to the upcoming chapters of this thesis.

Regression: In regression tasks, we typically assume a relationship between two variables of the form $\boldsymbol{y} = f(\boldsymbol{x})$. In general, the two variables can be multi-dimensional, and the objective is to find the function f that relates them.

For example, given satellite images, we may be interested in predicting the temperature of the soil. In this case, f would be a scalar function $f : \mathbb{R}^n \mapsto \mathbb{R}$ that takes the vector of pixel values $\boldsymbol{x} \in \mathbb{R}^n$ and outputs the temperature value y. Alternatively, we may be interested in inferring additional properties from the same images, such as the relative humidity, the population density, the level of pollution, etc. In this case, the output is also a vector $\boldsymbol{y} \in \mathbb{R}^m$, and $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a multi-dimensional function. We find interesting regression problems in a wide range of fields, such as sociology (e.g., annual salary as a function of years of working experience), psychology (e.g., perceived happiness relative to wealth), finance (e.g., housing market prices depending on socioeconomic factors) or in (quantum) physics and chemistry. In Chapters 7 and 8, we consider regression tasks related to the biophysical characterization of diffusion processes. The ML tasks involved in Chapters 4 and 5 also fall

within the regression type, although, in these cases, they are components of a larger ML setup, rather than the main objective.

In practice, we can neither optimize over the set of all possible functions nor over the entire domain of $\boldsymbol{x}$. Instead, we resort to a finite data set comprised of tuples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ for which we aim to find a model that maps every input $\boldsymbol{x}_i$ to its corresponding target $\boldsymbol{y}_i$. The regression model encoding f is usually predefined up to some parameters, which imposes certain conditions or assumptions on f. Among the simplest models, we can assume a linear relationship between the input and the output, characteristic of linear regression. From there, the model can be extended to increasingly more complex functions, as detailed in Section 2.2.2.

Classification: In classification tasks, the objective is to learn a function f that assigns an input $\boldsymbol{x}$ to a certain category or label y among a finite set of labels. Unlike in regression tasks, these categories do not need to have a numerical meaning and we typically assign them an integer index to encode them.

For example, given some movie reviews, we may be interested in telling apart the positive ($y = 0$) from the negative ($y = 1$) ones. This is a *binary classification* task in which the goal is to learn the function $f : \mathbb{R}^n \mapsto \{0, 1\}$, where $\boldsymbol{x} \in \mathbb{R}^n$ is an encoding of the text. We may be further interested in finding the genre the movies belong to among k options, with $f : \mathbb{R}^n \mapsto \{0, \dots, k-1\}$, which would correspond to a *multi-class classification* task. We encounter classification tasks across multiple fields, and they have served as benchmark tasks for the development of new ML algorithms and models for years. Some prominent examples are the hand-written digit classification task with the MNIST data set [41], featuring a class for every digit from zero to nine, or the classification of various objects using the CIFAR [42] or ImageNet [43] data sets, containing from 10 to 1,000 classes. In Chapters 7 and 9 we respectively tackle multi-class and binary classification tasks related to the study of sequential data.

While regression and classification tasks stand as the most common and paradigmatic examples in the field of ML, the zoo of possible tasks is both varied and extensive. Beyond these archetypal tasks, we encounter multiple examples that illustrate the versatility of ML techniques, such as text translation, text transcription from images or sounds, imputation of missing values (e.g., image reconstruction), anomaly detection, and denoising, among others.

2.1.2 Types of learning

Another crucial factor in the learning process is data, and its accessibility often determines the types of learning we can consider. The notions of task, as presented in the previous section, and data are intertwined: certain tasks can only be solved if sufficient data is available and richer data enables the seamless transition between tasks. In the ML field, we typically refer to data in terms of a data set $\mathcal{D}$, containing a finite amount of data instances often called *data points* or *examples* $\boldsymbol{x}_i$. The data set may exclusively contain the data instances, $\mathcal{D} = \{\boldsymbol{x}_i\}$, or they may also be accompanied by predefined labels or targets $\boldsymbol{y}_i$, forming tuples $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$. In some cases, the data points can be organized into a *design matrix* $\boldsymbol{X}$, formed by stacking $\{\boldsymbol{x}_i\}$ either row- or column-wise.

Each element of every data point $\boldsymbol{x}_i$ is known as a *feature*, and it is a descriptor of a specific aspect of the example. Selecting the right features to characterize the object of interest can be challenging: too few might fail to capture all relevant aspects,

whereas too many can lead to spurious correlations that interfere with the conclusions drawn from the data. Additionally, the data can be arbitrarily processed and transformed, for instance, subtracting the mean of every feature across the data set $\mathcal{D}$ from each data point prior to any further analysis. Determining the right data representation is a central problem in ML, as it is essential to perform any task, and it is the core of the field of representation learning. In Chapter 9, we delve into this topic, exploring different methods to represent stochastic trajectories of internet users browsing an e-commerce website, and extracting meaningful information from the engineered features.

In general, the type of available data effectively defines the types of learning our model can be faced with. These are usually divided into three categories: supervised, unsupervised, and reinforcement learning.

Supervised learning: Supervised learning can be understood as a generalized notion of regression and classification. It refers to ML algorithms that learn from *labeled* data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$. There exist various approaches to supervised learning, spanning from statistical methods to classical ML and DL. In this thesis, we employ supervised learning throughout Chapters 7 to 9 to address regression and classification tasks of stochastic trajectories. In most of these cases, substantial amounts of data are required for the training process, which entails the accurate labeling of the data. This is usually considered one of the most significant drawbacks of supervised learning, as obtaining perfectly matched labels are not always feasible or may require the manual addition by humans.

Unsupervised learning: While labeled data may be scarce, we often have access to large amounts of raw unlabeled data $\mathcal{D} = \{\boldsymbol{x}_i\}$. In this case, we can employ *unsupervised learning*, which refers to ML algorithms that learn from unlabeled data. Unsupervised learning can either be used for preliminary pre-processing steps, such as dimensionality reduction, or for representation learning, as in data clustering. Furthermore, it can be used to learn the underlying probability distribution of the data and generate entirely new examples. In Chapter 8, we use unsupervised learning to identify clusters in our predictions, which are related to different diffusive states of the particles. In Chapter 9, we explore different unsupervised dimensionality reduction techniques in our feature analysis. Finally, we show how to learn the data distribution from text to write new pieces in our open ML course and our tutorial about language modeling, accessible in Refs. [38] and [44], respectively.

Reinforcement learning. In contrast to the two previous types of learning, some ML do not rely on a fixed data set $\mathcal{D}$. For instance, in RL, we usually do not even have a data set available at all from the beginning. Instead, the learning algorithm interacts with an *environment* in a feedback loop to accomplish a given task. The data set is gradually shaped as the learning system collects experiences derived from these interactions. Different RL algorithms manage the collected data differently, although it is always leveraged to achieve the objective task. We employ different RL algorithms in Chapters 4 and 5, which are thoroughly introduced in Section 2.3.

2.1.3 Learning as an optimization task

The final components of the learning process are the performance measure and the model. The model encodes the structure of the resulting program, and it is typically a function of the data, $f(\boldsymbol{x})$, whose output depends on the task. For example, it can be a class from a discrete set of possible classes in a classification task, or a complex-valued tensor in a regression task. Finding the function that provides the best mapping between the data and the desired outcome for a specific task is at the heart of ML. The model is usually characterized by a set of parameters $\boldsymbol{\theta}$, and all its possible parametrizations form the set of functions known as the *hypothesis space*.

The performance measure objectively quantifies the model's performance in a given task. For example, in a classification task, we may consider the *accuracy*, which is the proportion of examples for which the model outputs the correct result. We can obtain the same information with the *error rate*, which is the proportion of missclassified examples. In a regression task, we may consider the absolute difference between the model's output and the desired value averaged over all data points, known as the mean absolute error (MAE).

We distinguish between *metrics* and *loss functions*. The former are performance measures that provide valuable information but may not be smooth, such as the accuracy. Conversely, the latter may not be as informative, but they are always differentiable, such as the MAE. Typically, loss functions quantify the errors performed by the model such that smaller losses translate into better models. For example, in Chapters 4, 7 and 8, we use the MAE as loss function for the regression tasks tackled therein, defined as

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n}\sum_{i}^{n} |y_i - f(\boldsymbol{x}_i)| \,, \tag{2.1}$$

where n is the size of the data set and y_i is the expected value for each sample $\boldsymbol{x}_i$. For the classificationt asks involved in Chapters 7 and 9, we use the cross-entropy loss function:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n}\sum_{i}^{n} \boldsymbol{y}_i^T \log(f(\boldsymbol{x}_i)) \,. \tag{2.2}$$

In these cases, $f(\boldsymbol{x}_i)$ is a vector containing the probability that the data point $\boldsymbol{x}_i$ belongs to each of the possible classes, and $\boldsymbol{y}_i$ is the *one-hot-encoding* vector of the true class, e.g., $\boldsymbol{y}_i = [0,0,1,0]^T$ encodes the third class in a four-class classification problem. The absolute value of $\mathcal{L}_{\text{CE}}$ does not provide a clear idea of the model's performance, unlike the accuracy, but it is a smooth function in its domain.

Machines "learn" by minimizing the loss function $\mathcal{L}$, over the training data set, i.e., the data available during the learning process. Formally, the objective is to find the optimal model parameters $\boldsymbol{\theta}^*$ in the hypothesis space that minimize $\mathcal{L}$. The minimization is usually performed by gradient-based techniques, hence the emphasis on the differentiability of $\mathcal{L}$. Therefore, learning becomes an optimization process.

2.1.4 Capacity, overfitting and underfitting

The central challenge in ML is to ensure the model can *generalize* well. This means that the resulting model should perform effectively on new data, never seen during the learning process. While learning, also known as training, our algorithm has access to the training data, as briefly introduced in the previous section. This allows us to quantify the performance over the so-called training set, obtaining a training error. To assess the generalization capabilities of the model, we hold out some data

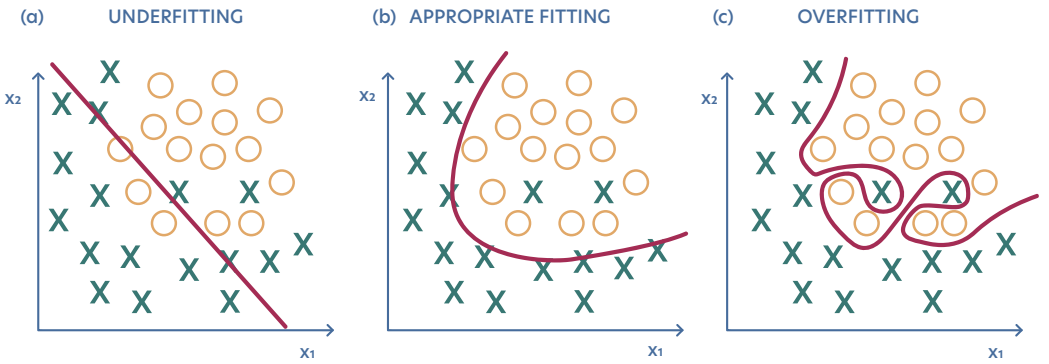

FIGURE 2.2: **Schematic depiction of under- and overfitting.** (a) When the model capacity is too low, it cannot fit the training data. (b) When the model capacity is adequate for the task, the fitting is optimal. (c) When the model capacity far exceeds the complexity of the task, the model overfits, resulting in a low training loss at the cost of poor generalization.

from the training process, forming a *test set*, and measuring the performance on this previously unseen data. This provides a test or generalization error. What distinguishes ML from mere function fitting is that in ML the goal is to keep both the training and test errors low, ensuring the proper generalization of the model.

Given that the parameter optimization of our model is performed in the training set, the expected test error is higher or equal than the expected training error. Their difference is known as the *generalization gap*, and it persists even when the training and test data are generated by identical probability distributions. Indeed, it may only disappear in the limit of infinite data. Hence, the overall performance of an ML model can be evaluated considering two factors: the training error and the generalization gap, which should both be as small as possible.

These two factors are related to two main challenges of ML: overfitting and underfitting. In short, underfitting occurs whenever the training error is not low enough, meaning that the model is unable to successfully develop the task. Overfitting is characterized by a large generalization gap, which is often the result of a low training error combined with a high test error. In these cases, the generalization is hindered by the model learning the specific details of the training data, rather than capturing the main patterns. Fig. 2.2 shows a schematic depiction of both phenomena.

The tendency of the model to under- or overfit can be tuned by changing the model's *capacity*. The capacity can be loosely understood as the measure of a model's ability to fit a wide variety of functions [39]. Models with low capacity may be unable to fit the training set, resulting in underfitting (Fig. 2.2(a)). On the other hand, models with a capacity much higher than required for the task tend to overfit (Fig. 2.2(c)), typically adjusting to the particularities of the training set, which may not be true for the general distribution. The challenge is to find the capacity that is "just right" for the task (Fig. 2.2(b)).

To address this challenge, the data set can be further split to include a *validation set*, resulting in three subsets: training, validation and test sets. The validation set serves as pseudo-test set to evaluate the model's generalization capabilities and adjust its *hyper-parameters*, which may impact its capacity. For example, the capacity can be controlled by constraining the model's hypothesis space. In the case of

a linear regression model, the hypothesis space contains all linear functions. Generalizing the linear model to include polynomial functions up to the k-th degree effectively increases its capacity, as demonstrated in our course [38]. Adjusting the maximum polynomial degree allows the fine-tuning of the model's capacity. In DL architectures, introduced in Section 2.2, the capacity can be adjusted by tuning the number of neurons and their connections.

Hence, the model learns from the data in the training set, and its hyper-parameters are tuned based on the generalization gap between the training and validation sets. This process may involve multiple training iterations as the model is being adjusted. Finally, the model's performance is evaluated in the test, providing a real measure of its performance on unseen data.

2.1.5 Regularization

In the previous section, we have shown that adjusting the model's capacity can improve its generalization. In general, every modification of the model with the goal to improve generalization, even if it leads to higher training error, is categorized as a *regularization* technique.

Regularization can be conceptualized in terms of Occam's razor or the principle of parsimony, which state that among competing hypotheses that explain known observations equally well, one should choose the simplest one [39]. Besides its philosophical motivation, adhering to this principle often yields better results in practice. Among models with comparable training performance, the simplest model tends to generalize the best [45].

A common approach to enforce these principles is by restricting the model's capacity. However, rather than enforcing it beforehand, we can introduce a penalty in the training loss function that is proportional to the magnitude of the model's parameters, known as *weight decay* [46, 47]. This kind of penalty ensures that an increase in model complexity only occurs when justified by a significant performance improvement. A widely used weight decay technique is L^2 regularization, which introduces a term proportional to the modulus square of the model parameters to the training loss function:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{train}} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}\,, \tag{2.3}$$

where λ is a hyper-parameter that adjusts the relative strength of the regularization term. We use this kind of weight decay to train the DL models in Chapters 7 to 9. Notably, L^2 regularization is efficiently implemented as the gradient of the regularization term is directly proportional to the parameter vector itself.

Nevertheless, regularization techniques extend beyond adjusting the model's capacity. One example is *early stopping*, which consists on halting the training process whenever the validation loss consistently shows signs of increase, indicating a potential overfitting. Shown to be highly effective [48], we leverage early stopping to train the DL models in Chapter 9.

Other regularization techniques target the way that models process the training data to enhance their generalization capabilities as well as to make them robust to the inherent nuances of real-world applications. *Data augmentation* consists on generating additional synthetic data by applying diverse affine transformations to every data point [49]. For example, in computer vision tasks, images can undergo rotations, stretches, crops, zooms, and more. This strategy not only expands the training data set, but also makes the model robust to such kind of transformations common in real-world scenarios. Another technique, *dropout* [50], omits randomly selected

features from every data point during the training process. This concept can extend beyond data points to every part of the model itself, such as between layers in DL architectures. By randomly removing bits of information that propagate through the model, dropout prevents it from relying on specific patterns in the training data and improves its resilience to incomplete information. In Chapters 7 and 8, we use both data augmentation and dropout to train the DL models. The data augmentation is performed by adding different kinds of noise to the samples during training.

2.2 Deep learning

Deep learning (DL) stands as a specialized branch of ML methods based on representation learning. As alluded to in the previous sections, representation learning comprises the methods that automatically discover the appropriate data representations to conduct a task. DL methods adopt a hierarchical structure featuring multiple levels of representation. These emerge from the composition of simple yet non-linear modules that transform the representation at one level into a representation at a higher, more abstract level. Through the composition of enough such transformations, DL models can learn very complex functions [51–53].

These abstract representations and transformations can be naturally represented by neural networks (NNs). The architecture of these NNs dictates the information flow, which is often organized in *layers* composed by neurons that do not share connections among themselves. The values of the neurons in a layer, commonly referred to as *activations*, encode a representation of the data. Then, the connections between neurons of a layer with those in adjacent layers encode the transformations applied to the representations.

2.2.1 Challenges motivating the use of deep learning

One of the main challenges of ML is the generalization to new examples with high-dimensional data. Indeed, many ML tasks become exceedingly difficult when the number of dimensions in the data is high, known as the *curse of dimensionality* [39].

The curse of dimensionality imposes a significant statistical challenge, primarily stemming from the vast number of potential data configurations far surpassing the available examples. For instance, consider the task of learning the probability distribution of the data. A straightforward approach consists on build a histogram partitioning the space into a grid. Given a new data point $\boldsymbol{x}$, it can be assigned a probability density based on the grid cell it belongs to within the histogram. While this approach may be effective for low-dimensional data, there is an exponential decay in the number of samples per grid cell with the data dimensionality, resulting into sparse histograms with numerous empty cells. As discussed in our ML course [38], attempting to learn the probability distribution of the MNIST data set [41] composed of 28×28 images is effectively impossible with this method even in its binarized form (black or white pixels). The 784-dimensional data yield approximately $2^{784} \approx 10^{236}$ possible configurations. Thus, even if every atom in the observable universe served as a training sample (estimated around 10^{80}), the resulting histogram would remain extremely sparse assigning null probability almost everywhere. Remarkably, DL models overcome the curse of dimensionality, showcasing their ability to generate high-resolution coloured images and videos [54, 55].

Traditional ML algorithms typically impose local constancy or smoothness in the functions they learn, such that $f(\boldsymbol{x}) \approx f(\boldsymbol{x} + \varepsilon)$ for small ε. For instance, the k-nearest neighbours [56, 57] or random forest (RF) [58] algorithms partition the space into constant-valued regions, akin to the previous histogram example. Nevertheless, these approaches usually require order $O(m)$ training examples to discern $O(m)$ regions in the space. This limits their capacity to learn complex functions with more distinct regions than learning examples and generalize well in high-dimensional data. In contrast, DL models impose additional explicit or implicit assumptions, also known as biases, to the functions they represent, which introduce further dependencies between distinct regions. For instance, a typical assumption is that the data follows a distribution generated by the composition of factors. This allows DL models to distinguish up to $O(\exp(m))$ regions with only $O(m)$ training examples [52, 53], enabling their generalization in high-dimensional data. Introducing further task-specific biases can improve generalization even further. For instance, designing DL models that respect the symmetries of the problem, such as translational or permutational invariance in convolutional neural networks (CNNs) and transformers, respectively, introduced in the upcoming Section 2.2.2.

Many ML scenarios only present relevant variations in low-dimensional manifolds, despite featuring vast high-dimensional spaces. In the ML terminology, manifolds refer to connected sets of points that can be effectively approximated with a small number of degrees of freedom in a higher-dimensional space [39]. Typically, the entire space consists of mostly invalid inputs, except for a few manifolds containing a small subset of points. Meaningful variations in the target function occur exclusively either along these manifolds or in their intersections. To illustrate it, let us recover the MNIST example of images of hand-written digits. Among all the possible pixel combinations, only a few yield sensible images where a digit can be recognized. Indeed, randomly sampled pixels resemble static noise, which deviates significantly from the highly structured images encountered in real life and practical ML applications. Intuitively, we can visualize the manifolds of all possible images for every digit, and meaningful changes in the output function happen when transitioning from one manifold to another, as the image label changes accordingly. DL models excel at capturing and efficiently navigating such low-dimensional manifolds [59].

2.2.2 Deep learning architectures

In this section, we present the different NN architectures implemented across the upcoming chapters of this thesis.

Fully-connected neural network

Fully-connected NNs, also called feedforward NNs or multi-layer perceptrons represent the fundamental architecture in DL. These networks are arranged in a sequence of layers of neurons, where connections exist only between neurons in adjacent layers, as illustrated in Fig. 2.3(a).

We distinguish three types of layers. The *input layer* contains the data values, such as pixel values of an image, or the position coordinates of a moving particle. The *output layer* contains the final value, which could be a real-valued vector for a regression task, or the encoding of a set of possible classes in a classification task, for instance. In practical applications, we mostly interact with the input and output

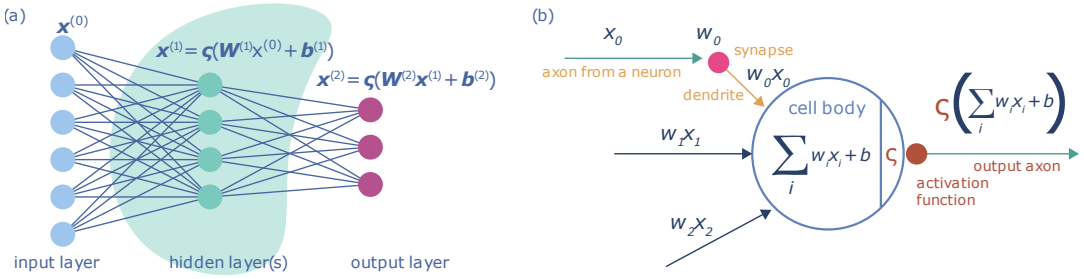

FIGURE 2.3: **Schematic representation of a fully-connected NN.** (a) Typical fully-connected NN architecture with a single hidden layer. (b) Schematic representation of a single neuron, showing the computation performed within and the information flow.

layers. Consequently, all the layers between them are known as *hidden layers*, which conduct additional computations to produce the desired result in the output layer.

Individual neurons perform simple calculations based on the signals received from the neurons in the preceding layer, as illustrated in Fig. 2.3(b). Typically, a neuron conducts a linear transformation followed by a non-linear *activation function* ς of the form:

$$z = \boldsymbol{w}^T \boldsymbol{x} + b \tag{2.4}$$

$$x = \varsigma\left(z\right) . \tag{2.5}$$

Here, $\boldsymbol{x}$ denotes the values or activations of the neurons in the preceding layer, and the connection strength with each neuron in that layer is captured by the parameter vector $\boldsymbol{w}$, commonly known as the *weights*. The neuron incorporates a *bias* b, and the resulting value of the linear transformation z is known as the *logit*. Finally, the resulting activation x of the neuron is determined by applying the non-linear activation function ς to the logit. This activation is then transmitted to the neurons of the subsequent layer, where an analogous computation is performed.

The activation function can take the form of any non-linear function. Frequently, functions that are both computationally efficient and differentiable are preferred. Two commonly used choices include the rectified linear unit, often abbreviated as ReLU:

$$\varsigma(z) = \max\left(0, z\right) , \tag{2.6}$$

and the hyperbolic tangent:

$$\varsigma(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} . \tag{2.7}$$

In this thesis, we extensively use the former in all DL models, and the latter in Chapter 5. While these activation functions are common for hidden layers, output layers typically have activation functions specific to the task. In classification tasks, the output layer provides a probability distribution over the possible layers, which is achieved with a *softmax* activation function:

$$\varsigma(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}} , \tag{2.8}$$

where z_i denotes the logit of the i-th neuron in the layer. In regression tasks where the output is bound within a known range, the sigmoid activation function is very

convenient, as it can be scaled and offset to match the range:

$$\varsigma(z) = \Delta \frac{1}{1+e^{-z}} + c\,, \tag{2.9}$$

where Δ and c are a scale factor and offset pre-determined by the task. We employ softmax and scaled sigmoid activation functions in Chapters 5, 7 and 8.

The activation of all neurons in layer ℓ can be computed in parallel by organizing the weights and biases of each neuron into a weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$, and then applying the activation function element-wise:

$$\boldsymbol{x}^{(\ell)} = \varsigma\left(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right). \tag{2.10}$$

In typical NN diagrams, such as the one in Fig. 2.3(a), the connections between fully connected layers encode such a linear transformation. Consequently, these layers are also commonly referred to as *linear layers*, despite the posterior non-linear activation assumed to occur in the neurons themselves.

We employ small fully-connected NNs in Chapters 4, 5 and 9. However, the modular nature of NNs enable the combination of different kinds of layers in the same architecture. Fully-connected NNs serve as key components to other architectures, typically used in the final parts to produce the output layer, as we do in Chapters 7 and 8.

Convolutional neural network

Convolutional neural networks (CNNs) [60] are a special kind of NNs with restricted connectivity between neurons in adjacent layers. Importantly, the majority of the weights are shared, forming the so-called *filters* or *kernels* of a fixed size. Every filter is characterized by a learnable weight pattern that is replicated along the entire preceding layer. This design leads to a highly local connectivity as well as a reduction in the number of weights compared to the fully-connected NNs, as illustrated in Fig. 2.4(a). Notably, the number of weights is independent of the layer size, which includes the size of the data, and it is solely determined by the number of filters and their size.

The outcome is akin to convolving these filters with the activations from the previous layer, thereby giving the network its name. We present an illustrative representation of the activations resulting from applying a 2-dimensional filter in Fig. 2.4(b). Typically, the activations of every individual filter are combined together such that the following layer receives as many activations as filters. In practice, the different filters typically specialize in recognizing specific patterns, such as the upper-left corner in the example from Fig. 2.4(b). The activation of the filter is maximal whenever it encounters the same pattern in the preceding layer's activations. While filters are typically small relative to the size of the data, and thus are restricted to detect simple and small patterns, the composition of multiple such convolutions enables the detection of increasingly complex patterns [42].

CNNs usually feature different layer combinations, and a prominent technique is the incorporation of *residual paths* [61]. Rather than sequentially processing the data directly from one layer to another, we establish two parallel paths for the information to propagate through the network: a convolutional path, where the convolutions are performed, and a so-called *identity* path, where the operations are kept at a minimum. The identity path allows the unrestricted information flow through the network, which enables the use of deeper architectures [62]. Every few layers,

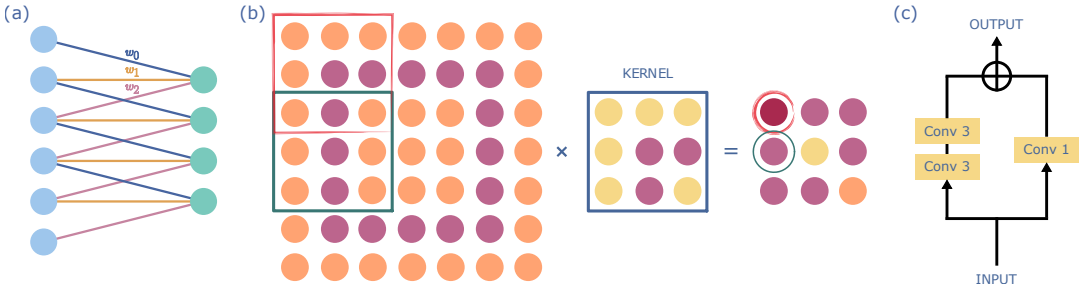

FIGURE 2.4: **Convolutional neural networks.** (a) Schematic representation of a single filter of size 3 with weights $\boldsymbol{w} = [w_0, w_1, w_2]$. The green neurons contain the activations of the filter replicated along the previous layer. (b) Illustration of a 2-dimensional convolution. The filter or kernel, is a 3×3 weight matrix that scans the input with stride 2, meaning that there is a separation of 2 points between consecutive steps, as exemplified by the red and green squares. The filter activation is maximal when the input presents a pattern that resembles its own. In this case, it is in the red square. (c) Schematic representation of a residual block. The input data follows two parallel paths: the convolutional path (left) with two consecutive convolutions, and the identity path (right), which performs a convolution with a 1-dimensional filter to match the output dimension from the convolutional path. The outcome of both paths is finally added together.

the outcome of both paths are combined, and we call the collection of layers between consecutive additions a *block*. We show an illustration of the residual blocks employed in Chapters 7 and 8 in Fig. 2.4(c). The convolutional path contains two consecutive convolutional layers with filters of size 3 and an activation function in between. In contrast, the identity path involves a single convolution with filters of size 1, capable of only rescaling the data. However, it serves as a mechanism to match the shape of the convolutional path for the posterior addition. At the end of the block, there is another activation function following the addition.

Recurrent neural network

Recurrent neural networks (RNNs) are class of NNs designed to process data sequentially. RNNs are autorregressive models, meaning that they have connections that form directed cycles, allowing them to preserve information from previous time steps as they advance through the data. This strcture allows them to capture temporal dependencies and learn patterns that evolve over time. Therefore, these networks are particularly well-suited to develop tasks that involve sequential data, such as time-series analysis or natural language processing (NLP), as we show in the example of Fig. 2.5.

At every time step, RNNs generate a *hidden vector* besides the regular output dedicated to the task, as illustrated in Fig. 2.5. The hidden vector is autorregressively fed back into the network to be processed along the input data in the subsequent time step. This mechanism enables the model to retain a memory of past time steps. In its simplest form, the RNN consists of a single linear layer that takes the current data $\boldsymbol{x}_t$ and the previous hidden vector $\boldsymbol{h}_{t-1}$ as input, and produces a new hidden vector $\boldsymbol{h}_t$ and an output $\boldsymbol{o}_t$ for the task.

$$\begin{bmatrix} \boldsymbol{h}_t \\ \boldsymbol{o}_t \end{bmatrix} = \varsigma \left(\boldsymbol{W} \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{bmatrix} + \boldsymbol{b} \right) . \tag{2.11}$$

Here, the hidden vectors are concatenated with the data and the output in flattened vectors. Notice that the model parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ remain constant with time.

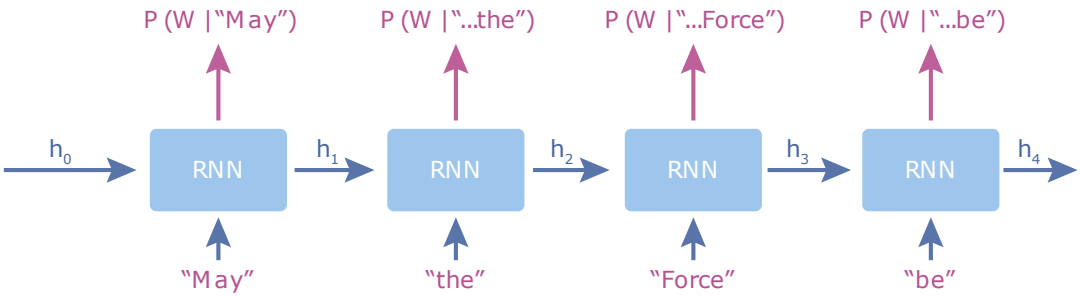

FIGURE 2.5: **Pictorial representation of an RNN.** The RNN sequentially processes the input data "May the Force be". At every step, it outputs the probability that word "W" is the next word conditioned the all the previous words. Additionally, the RNN outputs a hidden vector $\boldsymbol{h}_t$ that is used in the next time step, together with the next bit of input data, to perform the following prediction. The RNN carries the memory of the past time-steps in the hidden vectors that are fed autorregressively at every time-step.

While traditional RNNs provide a mechanism for capturing sequential patterns, they often face challenges in learning long-term dependencies. To address this issue, more advanced RNN architectures, such as long short-term memories (LSTMs) [63] and gated recurrent units (GRUs) [64], have been developed. LSTMs introduce memory cells and gating mechanisms that allow the network to selectively store and retrieve information over longer sequences. GRUs incorporate similar simpler gating mechanisms to control the flow of information, providing a balance between model complexity and effectiveness in capturing long-range dependencies. While both yield superior performance, there are no evident differences between both models [65]. In Chapter 7, we implement both LSTMs and GRUs, and, given no significant performance differences, we choose to work with GRUs due to their lower complexity. Conversely, we extensively employ LSTMs in Chapter 9.

Transformer

Transformers are a type of NN architecture that relies on a self-attention mechanism to capture dependencies between the different parts of the input data [66]. The self-attention, also known as scaled dot-product attention, allows the model to weigh the importance of different elements in the input data point relative to a particular element. These elements are features of the input data, which can range from the value at every time-step in a time series, to pixel patches in images. The model associates three vectors to every element: a key, a query, and a value, which are stored in the $\boldsymbol{K}$, $\boldsymbol{Q}$, and $\boldsymbol{V}$ matrices respectively. These vectors are typically extracted by processing every element with a different shallow trainable NN for each. The query vector of every element is compared with the key of the rest via a dot product to measure how much weight should be put in those relative to the current element. The value associated to this element is the weighted sum of the values based on the query-key similarities. All the similarities and weighted sums can be performed at once with the following expression:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V\,, \tag{2.12}$$

where d_k is the size of the key vectors. This mechanism allows transformers to capture patterns at all scales in the input data, unlike CNNs that are limited to local

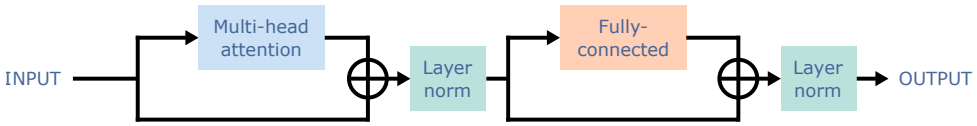

FIGURE 2.6: **Transformer encoder architecture.** The transformer encoder block is formed by two residual components followed by a layer normalization. The first one performs a multi-head attention on the input, and the second one applies an element-wise fully-connected NN, meaning that every element receives the same transformation.

correlations, and RNNs that struggle with long-range correlations.

The original transformer architecture consists of two main parts: an encoder and a decoder. Nevertheless, they can be used as elements of other DL architectures, as well as in encoder- and decoder-only models that can tackle different tasks. In this thesis, we focus on the encoder, illustrated in Fig. 2.6 as originally introduced in [66], which we use in Chapter 8 in combination with CNNs. The encoder block starts with a multi-headed attention layer in parallel with an identity path, similar to the residual networks from Fig. 2.4(c). The multi-headed attention layer performs multiple parallel executions of the self-attention mechanism from Eq. (2.12) with different parameters and concatenates the outputs together. The output is added to the identity path and the result is normalized. Finally, the result is processed by a point-wise fully-connected NN with the same residual scheme, which applies the same transformation to every element in the upcoming data. We refer to our hands-on live tutorial [44] for a thorough introduction on the transformer decoder and decoder-only language models, such as generative pre-trained transformers (GPTs) [67].

2.3 Reinforcement learning

Among the three types of learning introduced in Section 2.1.2, reinforcement learning (RL) [68] stands out as the most unique in multiple aspects. While typical supervised or unsupervised ML scenarios involve extracting significant patterns from the data to, e.g., infer labels, predict certain values or identify clusters, RL focuses on the idea of *learning strategies*.

The supervised learning framework can be envisioned as a scenario where a student learns from a teacher who possesses the knowledge to all correct answers within a given domain. However, this setup has inherent limitations, as the student can never surpass the teacher's expertise or tackle questions beyond its domain. To overcome these limitations, RL removes the figure of the teacher and allows the student explore and learn from the resulting experience. We refer to the student as the *agent*, as it can actively take actions. Just like us, humans, the agent learns from the interaction with an *environment*, understands the consequences of its actions, and formulates strategies to accomplish specific tasks.

Framing problems as games to discover strategies on holds great potential across multiple applications. This approach naturally fits control problems, although we can design games to accomplish more abstract tasks, from designing new quantum experiments [22] to faster matrix multiplication or sorting algorithms [32, 33]. In this section, we introduce the basic formalism of RL and the main algorithms used in Chapters 4 and 5.

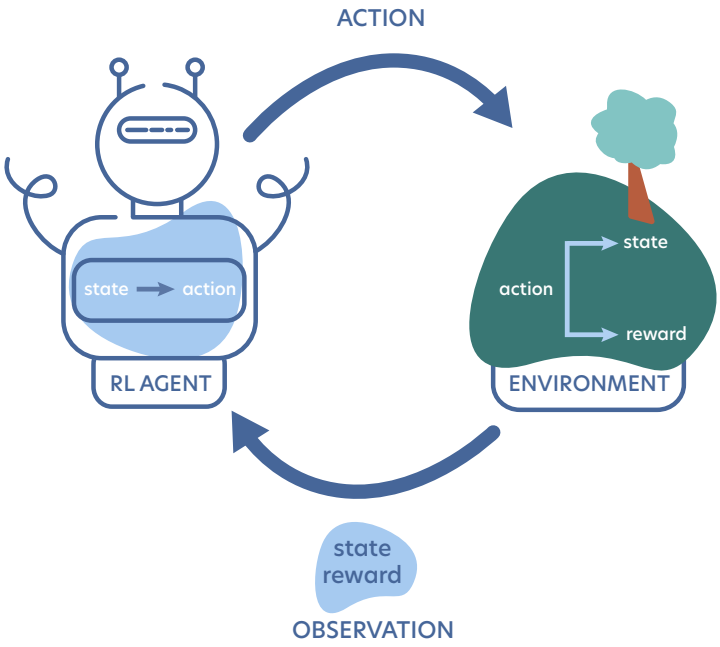

FIGURE 2.7: **Overview of the basic RL setting.** The agent receives an observation from the environment. Given the observation, it chooses to perform an action according to its policy. The environment determines the outcome of the action, and it returns an observation to the agent consisting of the new state and a potential reward.

2.3.1 Foundations of reinforcement learning

The general setting of any RL problem consist of two main elements: an *agent*, and an *environment* that it interacts with, as illustrated in Fig. 2.7. The environment contains all the information defining the problem at hand, e.g., the rules of a game, and it provides the agent with observations and feedback according to its *actions*. The environment defines the set of all possible *states*, $\mathcal{S}$, which can range from an empty set, in the case of a stateless environment, to a multi-dimensional continuous space. For example, these could be all the possible configurations of a board game or all the possible combinations of joint angles in a robot.

The agent can observe (sometimes only partially) the state s of the environment, and it can choose to perform an action a, which may include the possibility to remain idle. The action is chosen from the set of possible actions, $a \in \mathcal{A}$, defined by the environment and can be state-dependent. For instance, the action of pushing forward a pawn in chess is only possible if there is a free position in front of it. The actions may alter the state in which the environment is found, and they can have deterministic or stochastic outcomes. In the chess example, all the actions are deterministic. In contrast, in the case of a walking robot, the action to move forward may have different results: it can succeed in doing so, the robot may trip, or it may even remain idle with a certain probability due to a hurdle or malfunctioning. This information is encoded in the environment, and the agent may not have access to it.

Nevertheless, every time the agent performs an action, the environment provides it with an observation of the new state together with a feedback signal called *reward*, r. The reward can take any numerical value, and it may depend on the previous state, the new state, and the action that was taken. The main purpose of the agent

is to maximize the obtained rewards through its actions. Hence, the agent obtains higher rewards when accomplishing the objective task or progressing toward the goal, e.g., winning a game, while it might receive penalties when performing harmful or bad actions, e.g., losing a game.

More precisely, the agent aims to maximize the *discounted return*, and it is therefore the quantity that defines the RL task. The return is the weighted sum of the collected rewards. At a given discrete time t, the agent observes a state s_t, and performs an action a_t. This leads to a new state s_{t+1} and a reward r_{t+1}. The discounted return from time t is the weighted sum of future rewards until the final time T:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \tag{2.13}$$

where the discount factor $\gamma \in [0, 1]$ weights the rewards according to their temporal separation, penalizing those that are obtained far into the future. This concept draws inspiration from human psychology, and it mimics our daily observation that far-term rewards, even if high, are less desired than near-term ones, e.g., we favor procrastinating instead of reading this thesis. We can distinguish two limits: for a small discount factor, $\gamma \to 0$, the return becomes myopic and immediate rewards predominate over any other possible future ones. On the other hand, large discount factors, $\gamma \to 1$, result in equal weights for early and late rewards, which encourage long-term oriented strategies.

The central objective of any RL problem is to learn the *optimal policy*, π^*, that maximizes the discounted return. A policy, π, dictates which actions to take given the observations, and thereby defines the strategy followed by the agent. In general, the policy can take any form, as we show in forthcoming sections. For example, it can be a table assigning the best possible action to every possible state or an ML model that, given a state, provides a probability distribution over all the possible actions. In all cases, the learned policy is specific to the task, and it strongly depends on the reward function and the discount factor.

Markov decision processes

RL is mostly formulated with the underlying mathematical structure of Markov decision processes (MDPs). MDPs provide a general framework for modeling environments where a sequentiality notion exists between states. In such environments, the *Markov property* holds, meaning that the future is independent of the past given the present. In essence, this property asserts that the current state is a sufficient statistic, containing all the required information relevant to the possible evolution of the environment. In particular, the Markov property implies the absence of memory effects from previously visited states. Formally, at any time step t,

$$p(s_{t+1}|s_0, \ldots, s_t) = p(s_{t+1}|s_t). \tag{2.14}$$

Mathematically, an MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, G, \gamma)$, respectively denoting the state space $\mathcal{S}$, the action space $\mathcal{A}$, the *dynamics* p, the set of total returns G, and the discount factor γ. In this formalism, the return G, together with the discount factor γ, determines the objective, and p describes the environment dynamics,

$$p(s', r|s, a) = p(s_{t+1} = s', r_{t+1} = r|s_t = s, a_t = a)\,, \tag{2.15}$$

which corresponds to the joint probability of observing a new state s' and obtaining a reward r by performing action a in state s. In fully deterministic environments, $p(s', r|s, a)$ is either zero or one.

From Eq. (2.15) we can derive all the relevant information about the environment. For instance, *state-transition probabilities* are a central quantity in many RL algorithms:

$$p(s'|s, a) = \sum_r p(s', r|s, a) \,. \tag{2.16}$$

Furthermore, it allows us to determine the reward functions. In the most general form, the reward is jointly determined with the state s', as shown in Eq. (2.15).[1] However, in many cases, we may need to consider the expected rewards for state−action pairs and state−action−next-state triplets:

$$r(s, a) = \sum_r \sum_{s' \in \mathcal{S}} r\, p(s', r|s, a) \,, \tag{2.17}$$

$$r(s, a, s') = \sum_r r \frac{p(s', r|s, a)}{p(s'|s, a)} \,. \tag{2.18}$$

In the iterative interaction between agent and environment, the agent chooses the actions according to a policy. In its most general form, the policy is a mapping from state to the probability of performing each possible action. In the limit of deterministic policies, $\pi(a|s)$ is one for a single action and zero for the rest. During the learning process, the policy is improved based on the experience gathered from the interaction with the environment to achieve the goal. This interaction generates *trajectories* of the form

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots, s_T \,,$$

where all states, actions and rewards are random variables. This way, the agent performs a trajectory through the state-action space $\tau = a_0, s_1, a_1, \ldots, s_T$ with probability

$$p(\tau) = \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \,, \tag{2.19}$$

starting from an initial state s_0. We denote the discounted return associated to the trajectory as $G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$.

This entire formalism holds assuming the Markov property from Eq. (2.14), which implies that the environment is memory-less. However, we may encounter situations in which the environment has certain memory effects. In these cases, we may recover the Markov property by considering an extended state space that already includes the memory. In return, this implies that even deterministic Markovian dynamics on the full state space can give rise to non-deterministic and non-Markovian dynamics on the smaller state space.

Value functions and Bellman equations

As we mention in the previous sections, the goal in RL is to find the optimal policy π^* that maximizes the return, introduced in Eq. (2.13). Such a clear objective allows us to define *value functions* that estimate how convenient it is for the agent to be in a given state or to perform a certain action to accomplish the task. For instance,

[1] In stochastic environments, the reward can be inherently sampled from a probability distribution. Consider the game of blackjack: with the same hand (state) the action of settling may have different rewards depending on the opponent's hand (environment). Hence, the reward is stochastic.

consider the case in which we are looking for a treasure on a map. Being one step away from the treasure is, overall, much better than being ten steps away. However, not all actions in the close position are equally good, provided that one leads to the treasure but the others move away from it. This is quantified by the expected future return that the agent may obtain given the current conditions. However, given that the future rewards strongly depend on the actions that the agent will take, value functions are defined with respect to the policy.

The *state-value function*, $V_\pi(s)$, of a state s under the policy π is the expected return when starting at state s and following the policy π thereafter. We formally define it as

$$V_\pi(s) = \mathbb{E}\left[G_t | s_t = s, \pi\right] = \mathbb{E}\left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \middle| s_t = s, \pi\right] \tag{2.20}$$

In a similar way, the *action-value function*, $Q_\pi(s,a)$, is the expected return when starting at state s, performing action a, and then following the policy π:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}[G_t | s_t = s,\ a_t = a, \pi] = \\ &= \mathbb{E}\left[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a, \pi\right] \end{aligned} \tag{2.21}$$

The *advantage*, $A_\pi(s,a)$, is the additional expected return obtained by following an action a at state s, over the expected policy behavior:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)\,. \tag{2.22}$$

The value functions fulfill a recursive relationship that is exploited by many RL algorithms, which stems from the recursive nature of the return $G_t = r_{t+1} + \gamma G_{t+1}$. This allows us to write the state-value function $V_\pi(s)$ as a function of the next states

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[G_t | s_t = s, \pi] = \mathbb{E}[r_{t+1} + \gamma G_{t+1} | s_t = s, \pi] \\ &= \sum_a \pi(a,s) \sum_{s',r} p(s',r|s,a) \left(r + \gamma\, \mathbb{E}[G_{t+1} | s_{t+1} = s', \pi]\right) \\ &= \sum_a \pi(a,s) \sum_{s',r} p(s',r|s,a) \left(r + \gamma V_\pi(s')\right) \\ &= \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s, \pi]\,. \end{aligned} \tag{2.23}$$

We can do the analogous derivation for the action-value function $Q_\pi(s,a)$

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}[G_t | s_t = s, a_t = a, \pi] = \mathbb{E}[r_{t+1} + \gamma G_{t+1} | s_t = s, a_t = a, \pi] \\ &= \sum_{s',r} p(s',r|s,a) \left(r + \gamma\, \mathbb{E}[G_{t+1} | s_{t+1} = s', \pi]\right) \\ &= \sum_{s',r} p(s',r|s,a) \left(r + \gamma V_\pi(s')\right) \\ &= \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s, a_t = a, \pi]\,, \end{aligned} \tag{2.24}$$

from which the relationship $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a)$ becomes evident. These are the *Bellman equations* for the value functions, and they lie at the core of RL as they define the relation between the value of a state s and its successors s', recursively capturing future information.

These concepts introduce the notion of partial ordering between policies. A policy π is better than another policy π' if it yields a higher return. Hence, $\pi > \pi'$ if and only if $V_\pi(s) > V_{\pi'}(s)\ \forall s \in \mathcal{S}$. Therefore, the optimal policy π^* is such that it is better than or equal to all the other possible policies. [2] Hence, the optimal policy maximizes the value function. Taking the Bellman equations, Eqs. (2.23) and (2.24), π^* is such that

$$\begin{aligned} V_{\pi^*}(s) &= \max_a \mathbb{E}[G_t | s_t = s, a_t = a, \pi^*] \\ &= \max_a \mathbb{E}[r_{t+1} + \gamma V_{\pi^*}(s_{t+1}) | s_t = s, a_t = a, \pi^*] \\ &= \max_a Q_{\pi^*}(s,a)\,. \end{aligned} \tag{2.25}$$

Notice that in this new Bellman equation there is a maximization over the first action, as opposed to the expectation over actions from Eq. (2.23). This is because the value of a state under the optimal policy must be equal to the expected return for the best action. In a similar way, we can find the Bellman equation for the action-value function $Q_\pi(s,a)$ for an optimal policy π^*. Together, they define the set of the *Bellman optimality equations*:

$$\begin{aligned} V_{\pi^*}(s) &= \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_{\pi^*}(s')\right] \\ Q_{\pi^*}(s,a) &= \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} Q_{\pi^*}(s',a')\right] \end{aligned} \tag{2.26}$$

These equations fulfill

$$\begin{aligned} Q_{\pi^*}(s,a) &= \max_\pi Q_\pi(s,a) \\ V_{\pi^*}(s) &= \max_\pi V_\pi(s) = \max_a Q_{\pi^*}(s,a)\,. \end{aligned} \tag{2.27}$$

We can define the optimal policy $\pi^*(a|s)$ and action a^* at a given state s as:

$$\begin{aligned} \pi^* &= \arg\max_\pi V_{\pi^*}(s) \\ a^* &= \arg\max_a Q_{\pi^*}(s,a)\,. \end{aligned} \tag{2.28}$$

The optimal policy π^* corresponds to the deterministic choice of the best action a^* for a given state s according to the optimal action-value function $Q_{\pi^*}(s,a)$. Due to the recursive nature of the value functions, a greedy action according to V_{π^*} or Q_{π^*} is optimal in the long term.

The Bellman optimality equations Eq. (2.26) are, indeed, a system of equations with one for every state. In order to solve them directly, we need to explicitly use $p(s',r|s,a)$.[3] This is typically unknown and, therefore, we need additional methods to solve them, such as the ones we introduce in the following sections.

[2] The ordering operator is not always defined between policies. Two policies π, π' cannot be ordered iff $\exists\, s, s' \in \mathcal{S} :\ V_\pi(s) > V_{\pi'}(s),\ V_\pi(s') < V_{\pi'}(s')$. However, for MDPs there always exist an optimal policy π^* s.t. $\pi^* \geq \pi\ \forall \pi$ [68].

[3] Due to the maximization step in Eq. (2.26), this is a nonlinear optimization problem.

2.3.2 Value-based methods

In value-based RL, the goal is to obtain the optimal policy $\pi^*(a|s)$ by learning the optimal value functions, as in Eq. (2.28). Starting with initial estimation of the value function for every state, $V_\pi(s)$, or state−action pairs, $Q_\pi(s,a)$, they are progressively updated based on the experience gathered by the agent following its policy.

Since the value functions are defined with respect to a policy, we need to define a fixed policy for this family of algorithms. A common choice is an ε-greedy policy, with which the agent follows a deterministic greedy policy, e.g., choosing $a = \arg\max_a Q_\pi(s,a)$, and taking random actions with probability ε. This is a natural choice provided that the optimal policy is greedy with respect to the optimal value function. Hence, learning the value function for such policy provides us with the optimal one in the greedy limit ($\varepsilon \to 0$).

One of the most straightforward and naive approaches to learn the value function is to sample trajectories $\tau \sim p(\tau)$ (Eq. (2.19)), and then use the return G_t to update the value function estimation[4] for every visited state s_t along the way:

$$V_\pi(s_t) = V_\pi(s_t) + \eta(G_t - V_\pi(s_t))\,, \tag{2.29}$$

where η is a learning rate that controls the size of the update. We can do an analogous process for every visited state and action along the trajectory to learn $Q_\pi(s,a)$ instead.

However, with this approach we can only learn at the end of each trajectory, also known as *episodes*, which can be very inefficient in problems involving long episodes, or even infinite ones. On the contrary, temporal-difference (TD) algorithms exploit the recursive nature of the value functions, Eqs. (2.23) and (2.24), to learn at every time step:

$$V_\pi(s_t) = V_\pi(s_t) + \eta\left(r_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)\right)\,. \tag{2.30}$$

Notice that, while $V_\pi(s_t)$ is an estimate, $V_\pi(s_{t+1})$ is also an estimate. This is known as a *bootstrapping* method, as the update is partially based on another estimate. Nevertheless, it is proven to converge to a unique solution. The term in brackets is known as *TD error*.

The algorithm implementing Eq. (2.30) is known as TD(0), which is a special case of the TD(λ) algorithms [69]. The analogous algorithm for the action-value function is known as SARSA [70, 71]:

$$Q_\pi(s,a) = Q_\pi(s,a) + \eta\left(r + \gamma Q_\pi(s',a') - Q_\pi(s,a)\right)\,, \tag{2.31}$$

where we have recovered the notation s', a', r to denote the next state, action and reward. Replacing the term $Q_\pi(s',a')$ by an expectation over the next possible actions, such as $\sum_{a'} \pi(a'|s')Q_\pi(s',a')$, we obtain the expected SARSA algorithm [72]. If, instead, we take a maximization, as in Eq. (2.32) below, we obtain the Q-learning algorithm [73]. We devote the following Section 2.3.2 to the latter.

Q-learning

Q-learning is one of the most widely used TD algorithms due to its desirable properties [73]. Most of the TD algorithms that we introduce in the previous section

[4] The return is an unbiased estimator for the expectation $V_\pi(s_t) = \mathbb{E}[G_t|s_t,\pi]$ from Eq. (2.20). This is known as a sample update, as we only use a single sample to determine the expectation.

learn the value functions for their given policies, mainly ε-greedy policies. These include exploratory random actions that have an impact on the learned value functions. Therefore, the policy determines the result, and we must adjust ε during the training process to ensure their proper convergence toward the optimal value functions. However, Q-learning always learns the optimal action-value function regardless of the policy followed during the training.[5]

The goal is to directly learn the optimal Q-values, $Q_{\pi^*}(s,a)$, hence the name Q-learning, in order to obtain $\pi^*(s|a)$ by performing greedy actions over them, as in Eq. (2.28). We start by arbitrarily initializing our estimates $Q_\pi(s,a)\ \forall s \in \mathcal{S}, a \in \mathcal{A}$, which are typically stored in a table (see Section 2.3.2 for an implementation with NNs). Then, we sample trajectories $\tau \sim p(\tau)$ according to the policy to progressively update our estimates with the relation

$$Q_\pi(s,a) = Q_\pi(s,a) + \eta \left(r + \gamma \max_{a'} Q_\pi(s',a') - Q_\pi(s,a) \right) . \tag{2.32}$$

We illustrate the process in Algorithm 1.

Algorithm 1 Q-learning

Require: learning rate η, maximum time T, policy parameter ε
  Initialize $Q(s,a)\ \forall s \in \mathcal{S}, a \in \mathcal{A}$
  while not converged **do**
    Initialize s_0
    for $t = 0$ to $T-1$ **do**
      $\xi \leftarrow$ uniform$\in [0,1]$
      $a \leftarrow$ uniform a **if** $\xi \leq \varepsilon$ **else** $\arg\max_a Q_\pi(s,a)$ ▷ ε-greedy policy
      Move to next state s' and obtain reward r
      $Q(s,a) \leftarrow Q(s,a) + \eta \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$.
    end for
  end while
  return $Q(s,a)$ ▷ Optimal action-value function for all states and actions

This method is guaranteed to converge to the optimal action-value function as long as all possible state−action pairs continue to be updated. This is a necessary condition for all the algorithms that converge to the optimal behavior and it can become an issue for fully deterministic policies. However, with Q-learning, we can have an ε-greedy policy with $\varepsilon \neq 0$ that ensures that this condition is fulfilled.

The key element is that, while the policy determines which states and actions are visited by the agent, the Q-value update is performed over a greedy next action, as shown in Eq. (2.32). This way, the learned Q-values are those corresponding to the greedy policy over them, which is the one fulfilling the Bellman optimality equations Eq. (2.26).

Double Q-learning

Most of the TD algorithms suffer from a maximization bias that results in an overestimation of the Q-values, which can harm the performance. Specially, in Q-learning, we encounter two maximizations: one in the ε-greedy policy and one in the greedy

[5] Q-learning is an off-policy algorithm, which means that the policy it learns (optimal $\pi^*(a|s)$) is different from the one it follows in the training episodes. Algorithms like SARSA are on-policy, and learn the value function that corresponds to the policy with which they generate the training data.

target policy (Eq. (2.32)). This way, we use a maximum overestimated value (see below) to update the maximum Q-value, which corresponds to the greedy action taken by the policy, potentially incurring into a significant positive bias for $Q_\pi(s,a)$.

The maximization over next possible actions in Eq. (2.32) is a sample estimate for the maximum expected value $\max_{a'} \mathbb{E}[Q_\pi(s',a')]$. However, it is a positively biased estimator, provided that the sample estimate actually corresponds to the expected maximum value $\mathbb{E}[\max_{a'} Q_\pi(s',a')]$ [74]. In Ref. [68] they provide a simple example to develop intuition on the matter: suppose that the true Q-values for all actions in a state are zero and that our estimates $Q_\pi(s,a)$ are distributed around them taking positive and negative values. The maximum value is positive and, hence, it is an overestimation. The overestimation of the Q-values can prevent the algorithm from learning the optimal policy [75].

We overcome this issue with double Q-learning [76]. This way, instead of learning a single set of Q-values, we learn two: $Q^A_\pi(s,a)$, and $Q^B_\pi(s,a)$. However, in order to update one, we use the other to estimate the value of its corresponding next greedy action:

$$Q^A_\pi(s,a) = Q^A_\pi(s,a) - \eta \left(r + \gamma Q^B_\pi \left(s', \arg\max_{a'} Q^A_\pi(s',a') \right) - Q^A_\pi(s,a) \right) , \quad (2.33)$$

where A, B are interchangeable. This approach avoids using the same estimate to determine both the maximizing action and its value, yielding an unbiased estimate.

We learn both sets of values by randomly updating one at a time at every time step. The only additional difference with respect to standard Q-learning is that we take actions following an ε-greedy policy that combines the information of both $Q^A_\pi(s,a)$ and $Q^B_\pi(s,a)$, e.g., using their sum or mean. With double Q-learning, we overcome a major limitation of Q-learning at the price of doubling the memory requirements.

Double deep Q-learning

In Q-learning, as introduced so far, we need to explicitly store the Q-values, $Q_\pi(s,a)$, for every possible state−action pair. This approach allows us to find the exact optimal action-value function. However, it is only viable for small problems, as the memory requirement quickly becomes unfeasible for moderately large ones.

In these cases, we must rely on an efficient way to represent $Q_\pi(s,a)\ \forall s \in \mathcal{S}, a \in \mathcal{A}$. NNs are a prominent candidate to approximate the action-value function, as introduced in Ref. [77], with significantly less parameters than state−action pairs. Using DL models to learn the Q-values is known as *deep Q-learning* and the implemented NN is commonly referred to as a deep Q-network (DQN). DQNs take a representation of state in the input layer $\phi(s)$, and have as many neurons as possible actions in the output layer, which encode $Q_\pi(s,a;\boldsymbol{\theta})\ \forall a \in \mathcal{A}$. Here, $\boldsymbol{\theta}$ denotes the set of trainable parameters of the neural network. This way, the DQN provides the Q-value of all possible actions given a state.

Nevertheless, DQNs may become highly unstable when directly applying Algorithm 1 with an update rule for the network parameters:

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \eta \left(r + \gamma \max_{a'} Q_\pi(s',a';\boldsymbol{\theta}) - Q_\pi(s,a;\boldsymbol{\theta}) \right) \nabla_{\boldsymbol{\theta}} Q_\pi(s,a;\boldsymbol{\theta}) , \quad (2.34)$$

which is analogous to a regression problem in which we minimize the mean squared error between the target, $r + \gamma \max_{a'} Q_\pi(s', a'; \boldsymbol{\theta})$, and the prediction, $Q_\pi(s, a; \boldsymbol{\theta})$, through gradient descent. The instabilities are mainly due to correlations in consecutive observations along the trajectories, correlations between target and prediction, and significant changes in the data distribution due to small variations in the parameters. The latter happen because the agent follows an ε-greedy policy, and small changes in the parameters may change the actions that have the maximum Q-value for the states, abruptly altering the course of the trajectories. [6] We overcome these limitations with *experience replay* [78], and introducing a *target network*.

With experience replay, instead of learning at every time step, we store the experience gathered along the episodes in a memory, which keeps the information of every transition (s, a, r, s'). Then, once the agent has gathered enough experience, it replays a randomly sampled batch of transitions in its memory to compute the loss and update the DQN parameters. This way, the agent alternates between episodes to gather experience and replaying it to perform the learning process. This technique removes the correlation between training samples and mitigates the sudden changes in data distribution. Furthermore, it allows the agent to reuse the experience to prevent forgetting and re learning, which is particularly valuable when the experience is costly to obtain. For instance, if a robot receives severe damage, having a memory allows it to keep learning from the situation without receiving further injuries.

In order to remove the correlation between target and prediction, we consider a target network, which is a clone of the DQN that we update at a different rate. While we update the DQN parameters, $\boldsymbol{\theta}$, at every learning iteration, we only update the parameters of the target network, $\boldsymbol{\theta}^-$, copying $\boldsymbol{\theta}$ every few iterations. Then, we use it to predict the target term $\max_{a'} Q_\pi(s', a'; \boldsymbol{\theta}^-)$, hence the name of the network. This ensures that the prediction, $Q_\pi(s, a; \boldsymbol{\theta})$, and the target are uncorrelated. Additionally, we can go a step further and use the target network for double Q-learning (see Section 2.3.2) in order to prevent the DQN from overestimating the action-value function, as introduced in Ref. [79].

Thus, the overall implementation consists on gathering experience by following an ε-greedy policy on the Q-values, $Q_\pi(s, a; \boldsymbol{\theta})$. Then, the agent replays randomly selected transitions from the experience to compute the mean squared error between the target and the prediction, while using a target network to perform double Q-learning:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \left(r_i + \gamma Q_\pi \left(s_i', \arg\max_{a'} Q_\pi(s_i', a'; \boldsymbol{\theta}); \boldsymbol{\theta}^- \right) - Q_\pi(s_i, a_i; \boldsymbol{\theta}) \right)^2 , \tag{2.35}$$

where i denotes the index in a batch of n randomly sampled transitions from the memory. Then, we perform a gradient descent step over the loss in Eq. (2.35) to update $\boldsymbol{\theta}$. Finally, every few iterations, we update the target network $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$. We use double deep Q-learning in Chapter 4.

[6] Consider the case of two separate paths that lead to different treasures. We initialize the Q-values arbitrarily, and the ε-greedy policy mainly takes the path with the highest one, while casually following the other with small probability ε. However, if the second one leads to a bigger treasure, its Q-value will eventually become the highest, and the data distribution will suddenly change to mainly sample this path and casually take the other.

2.3.3 Policy gradient methods

The main goal of policy gradient algorithms is to obtain the optimal policy $\pi^*(a|s)$ by proposing a parametrized ansatz $\pi_{\boldsymbol{\theta}}(a|s)$ and optimizing its parameters $\boldsymbol{\theta}$ to maximize the objective. Hence, finding the optimal policy $\pi^*(a|s)$ is equivalent to finding the optimal set of parameters $\boldsymbol{\theta}^*$ that best approximates it $\pi_{\boldsymbol{\theta}^*}(a|s) \approx \pi^*(a|s)$. This parametrization can take several forms, such as a NN, and controlling the shape of the policy may allow us to leverage prior knowledge about the task to obtain better results. Furthermore, the policies are stochastic, which have a natural exploratory character and the flexibility to also approximate deterministic policies.

In order to optimize the parameters, we use an objective function O_π that we aim to maximize. This can be any figure of performance, such as the state-value function V_π, the action-value function Q_π, or the return G. Having continuous parametrized policies, the objective function changes smoothly with changes in the parameters, which allows us to compute their derivatives. We approach the optimization by a gradient ascent method: we compute the gradient of the expectation value $\nabla_{\boldsymbol{\theta}} \mathbb{E}[O_\pi|\pi_{\boldsymbol{\theta}}]$, and perform a small update of the parameters $\boldsymbol{\theta}$. The expectation value is taken over the trajectories τ sampled according to the policy (recall Eq. (2.19)).

Directly evaluating the gradient is not straightforward because it depends on the stationary distribution of the states, which is inaccessible in the setting considered here. Hence, it is difficult to estimate the effect of the policy update on the state distribution. However, the *policy gradient theorem* [80, 81] provides an analytical form for the gradient of the objective function that does not involve the derivative over the state distribution.

Policy gradient theorem: For any differentiable policy $\pi_{\boldsymbol{\theta}}(a|s)$ and objective function O_π, the gradient of its expectation value $\nabla_{\boldsymbol{\theta}} \mathbb{E}[O_\pi|\pi_{\boldsymbol{\theta}}]$ can be expressed in terms of derivatives acting exclusively on the logarithmic policy $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$. The term $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$ is often referred to as the *score function*.

To build some additional intuition on the above theorem, let us consider an example with the total return $G(\tau)$ as objective function (see [68] for an extended proof with $V_\pi(s)$). Thus, we are interested in maximizing the expectation value $\mathbb{E}[G|\pi_{\boldsymbol{\theta}}]$, which is performed over the trajectories $\tau \sim p_{\boldsymbol{\theta}}(\tau)$. We restate Eq. (2.19) to explicitly show the parameter dependence:

$$p_{\boldsymbol{\theta}}(\tau) = \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi_{\boldsymbol{\theta}}(a_t|s_t). \tag{2.36}$$

Therefore, the expectation can be written as

$$\mathbb{E}[G|\pi_{\boldsymbol{\theta}}] = \sum_\tau p_{\boldsymbol{\theta}}(\tau)G(\tau). \tag{2.37}$$

In order to take the gradient, let us first recall the property of logarithmic derivatives $\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}} = p_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}$, which we apply in the following derivation:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}] &= \sum_\tau G(\tau) \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\tau) \\ &= \sum_\tau G(\tau) p_{\boldsymbol{\theta}}(\tau) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau). \end{aligned} \tag{2.38}$$

Then, from Eq. (2.36), we see that the only dependence on $\boldsymbol{\theta}$ from $p_{\boldsymbol{\theta}}(\tau)$ is in the policy. Therefore,

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tau) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t), \tag{2.39}$$

which, combined with Eq. (2.38), we obtain the expression

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}] &= \sum_{\tau} p_{\boldsymbol{\theta}}(\tau) G(\tau) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \\ &= \mathbb{E}\left[G(\tau) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \middle| \pi_{\boldsymbol{\theta}} \right] \end{aligned} \tag{2.40}$$

The importance of the policy gradient theorem lies in the fact that it yields a closed form for the gradient as an expectation value. As a consequence, it can be estimated via Monte-Carlo sampling over different trajectories τ. Furthermore, the gradient of the objective function is independent of the initial state s_0, as it does not depend on the policy.

REINFORCE

The REINFORCE algorithm [82] is one of the most commonly used policy gradient algorithms and it uses the return as objective $O_\pi = G(\tau)$. [7] The main principle of REINFORCE is to directly modify the policy to favor series of actions within the agent's experience that lead to a high return. This way, previously beneficial actions are more likely to happen the next time the agent interacts with the environment.

Formally, we solve the optimization problem $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_\theta]$. We find $\boldsymbol{\theta}^*$ via an iterative update rule in which we compute the gradient $\nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}]$ and perform a gradient ascent step in its direction. In practice, we estimate it by sampling a batch of n trajectories or episodes $\tau \sim p_{\boldsymbol{\theta}}(\tau)$ to approximate the expectation value from Eq. (2.40). This way, at learning iteration k,

$$\Delta\boldsymbol{\theta}_k \approx \frac{1}{n} \sum_{i=1}^{n} G(\tau_i) \sum_{t=0}^{T_i-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{2.41}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \nabla \boldsymbol{\theta}_k, \tag{2.42}$$

where η is the learning rate. [8] We illustrate the procedure in Algorithm 2.

However, the trajectory sampling introduces significant fluctuations to the expected quantities that result in large training variances, which is a general problem of Monte-Carlo-based approaches. Some episodes may be quite successful whereas some others could be a complete failure with very low returns. Such high variance results into unstable policy updates, which increase the convergence time toward the optimal policy. A common technique to tackle this issue is to introduce a *baseline*

[7] In Section 2.3.1 we mention the optimal policy maximizes $V_\pi(s)\ \forall s \in \mathcal{S}$. Taking $V_\pi(s)$ as objective, the gradient is $\nabla_{\boldsymbol{\theta}} \mathbb{E}[V_\pi(s)|\pi_{\boldsymbol{\theta}}] = \mathbb{E}[Q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)|\pi_{\boldsymbol{\theta}}]$ (see [68]). In REINFORCE, G_t acts as an unbiased estimator of $Q_\pi(a_t, s_t)$ to find the optimal policy, since $Q_\pi(a_t, s_t) = \mathbb{E}[G_t|s_t, a_t, \pi_\theta]$ from Eq. (2.21).

[8] In some cases, it is beneficial to compute the expectation of the gradient as a weighted sum over the trajectory returns. In this case, rather than dividing by n, we divide by $\sum_\tau G(\tau)$, which makes the update rule independent of the scale of the returns. This approach disregards trajectories with zero return, which do not contribute to the gradient and would dilute the information, yielding very small updates.

Algorithm 2 REINFORCE

Require: learning rate η, number of trajectories n, maximum time T
Require: randomly initialized differentiable policy $\pi_{\boldsymbol{\theta}}(a|s)$
  while not converged **do**
    for $i = 1$ to n **do**
      Initialize s_0
      for $t = 0$ to $T-1$ **do**
        Take action $a_t \sim \pi_{\boldsymbol{\theta}}(a_t|s_t)$ and store $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
        Move to next state s_{t+1} and store reward r_{t+1}
      end for
      $G^{(i)} \leftarrow \sum_t \gamma^t r_{t+1}$
      $z^{(i)} \leftarrow \sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$
    end for
    $\Delta\boldsymbol{\theta} \leftarrow (1/n) \sum_i G^{(i)} z^{(i)}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \Delta\boldsymbol{\theta}$
  end while
  return $\boldsymbol{\theta}$ ▷ Optimal policy parameters

into the returns, which reduces the variance of the method without incurring any bias, and therefore *should always be used*.

In order to provide a better description of the baseline, let us first rewrite Eq. (2.40) in a more convenient way, and omitting the condition $\mathbb{E}[\cdot|\pi_{\boldsymbol{\theta}}]$ for the rest of the chapter:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}] &= \mathbb{E}\left[\left(\sum_{t'=0}^{T-1} \gamma^{t'} r_{t'+1}\right) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right] \\
&= \mathbb{E}\left[\sum_{t'=0}^{T-1} \gamma^{t'} r_{t'+1} \sum_{t=0}^{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \sum_{t'=t}^{T-1} \gamma^{t'} r_{t'+1}\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right],
\end{aligned} \tag{2.43}$$

where in the first equation we write the explicit form of $G(\tau)$. In the second equation we use the relation

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}] = \nabla_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{t'=0}^{T-1} \gamma^{t'} r_{t'+1}\right] &= \sum_{t'=0}^{T-1} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau_{t'}}\left[\gamma^{t'} r_{t'+1}\right] \\
&= \sum_{t'=0}^{T-1} \mathbb{E}_{\tau_{t'}}\left[\gamma^{t'} r_{t'+1} \sum_{t=0}^{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right] \\
&= \mathbb{E}\left[\sum_{t'=0}^{T-1} \gamma^{t'} r_{t'+1} \sum_{t=0}^{t'} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right],
\end{aligned} \tag{2.44}$$

where $\mathbb{E}_{\tau_{t'}}$ denotes expectation over trajectories up to time t'. Then, in the third line of Eq. (2.43), we rearrange the terms in the summations and we find the explicit form of G_t offset by a γ^t factor. In the final expression, it becomes clearer how past rewards

in the trajectories do not contribute to the gradient of the policy from a given time onward, which recovers the Markov property.

We can reduce the variance in the gradient by introducing a state-dependent baseline $b(s_t)$ in Eq. (2.43) such that

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_{\boldsymbol{\theta}}] = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \left(G_t - b(s_t)\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right]. \tag{2.45}$$

Any baseline is appropriate as long as it does not depend on the actions, which ensures the gradient estimation remains unbiased, given that

$$\begin{aligned}
&\mathbb{E}\left[b(s_t)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right] = \mathbb{E}_{\tau_t}\left[b(s_t)\, \mathbb{E}_{\tau_{t:T}}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right]\right] \\
&= \mathbb{E}_{\tau_t}\left[b(s_t) \sum_{a_t} \pi_{\boldsymbol{\theta}}(a_t|s_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \underbrace{\sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)}_{1} \underbrace{\sum_{\tau_{t+1:T}} p_{\boldsymbol{\theta}}(\tau_{t+1:T})}_{1}\right] \\
&= \mathbb{E}_{\tau_t}\left[b(s_t) \nabla_{\boldsymbol{\theta}} \underbrace{\sum_{a_t} \pi_{\boldsymbol{\theta}}(a_t|s_t)}_{1}\right] = \mathbb{E}_{\tau_t}\left[b(s_t) \cdot 0\right] = 0\,,
\end{aligned} \tag{2.46}$$

where $\tau_{t:T}$ indicates a trajectory from time t until the end T. We move from the second to the third line using the property of logarithmic derivatives, as in Eq. (2.38). Notice that the expectation remains unbiased even if the baseline depends on $\boldsymbol{\theta}$.

While the expectation is unaffected, the baseline can have a major impact in the variance. [9] Let us consider the case of a state-independent baseline. We can find the optimal baseline that minimizes the variance in the gradient for each parameter. In order to simplify the notation, let z_k and b_k be the k-th components of the score function $z_k = \partial_{\boldsymbol{\theta}_k} \log \pi_{\boldsymbol{\theta}}(a|s)$ and a state-independent baseline vector, respectively. Hence, the goal is to minimize the variance of the term $(G_t - b_k)z_k$, [10] which is the argument of Eq. (2.45). Formally, we aim to find $b_k^* = \arg\min_{b_k} \mathrm{Var}\left[(G_t - b_k)z_k\right]$, that is such that $\partial_{b_k^*} \mathrm{Var}\left[(G_t - b_k)z_k\right] = 0$. Therefore,

$$\mathrm{Var}\left[(G_t - b_k)z_k\right] = \mathbb{E}[((G_t - b_k)z_k)^2] - \mathbb{E}[G_t z_k]^2 \tag{2.47}$$

$$\partial_{b_k} \mathrm{Var}\left[(G_t - b_k)z_k\right] = -2\,\mathbb{E}[(G_t - b_k)z_k^2] \tag{2.48}$$

$$b_k^* = \frac{\mathbb{E}[G_t z_k^2]}{\mathbb{E}[z_k^2]}\,, \tag{2.49}$$

where in the first equation we have used Eq. (2.46) to remove b_k in the second term.

There are several other valid baselines that we can consider, besides the state-independent example above, with which may yield better results. For instance, an estimation of the value function $\hat{V}_\pi(s_t) \approx \mathbb{E}[G_t|s_t]$ is a common state-dependent baseline. This can either be learned, either directly from G_t or as we show in Section 2.3.4, or it can be estimated through sampling in self-critic schemes (see [83]). With such baseline, actions that lead to returns higher than expected with the current policy are reinforced, while those that lead to lower rewards are penalized. This is equivalent to weighting the score function by the advantage. Given that

[9] Recall that $\mathrm{Var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$. Hence, adding a term with null expectation does not affect the second term but it does have an impact on the first one $\mathrm{Var}[x - b] = \mathbb{E}[(x-b)^2] - \mathbb{E}[x-b]^2 = \mathbb{E}[(x-b)^2] - \mathbb{E}[x]^2$.

[10] In this case, we take the approximation $\mathrm{Var}\left[\sum_t X_t\right] \approx \sum_t \mathrm{Var}\left[X_t\right]$

$\mathbb{E}[G_t|s_t, a_t] = \mathbb{E}[Q_\pi(s_t, a_t)|s_t, a_t]$, from Eq. (2.21), subtracting a baseline $b(s_t) = V_\pi(s_t)$, we obtain the expectation of the advantage (recall Eq. (2.22)). Hence, $\nabla_{\boldsymbol{\theta}} \mathbb{E}[G|\pi_\theta] = \mathbb{E}\left[\sum_t \gamma^t A(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right]$. Directly estimating the advantage provides the least possible variance, see [84] for further reference on this matter.

Another common practice is to *whiten* the return. This consists of subtracting the mean of the return along all the time steps of a trajectory and dividing by its standard deviation $\bar{G}_t = (G_t - \mathbb{E}[G])/\sigma_G$. Since this is not exactly a baseline, this method *does* introduce a bias.

Deep REINFORCE

The parametrized policy $\pi_{\boldsymbol{\theta}}$ is a central quantity in policy gradient methods and it can take any form as long as it is differentiable with respect to its parameters. One of the most common approaches in discrete action spaces is to define action probabilities according to a *softmax* distribution:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{x(s,a)}}{\sum_{a' \in \mathcal{A}} e^{x(s,a')}}, \tag{2.50}$$

where $x(s, a)$ is the *action preference* for action a in state s.

The simplest way to define action preferences is through a set of linear parameters $\boldsymbol{\theta}$ applied to a feature representation of the state and action $\phi(s, a)$, such that $x(s, a) = \boldsymbol{\theta}^T \phi(s, a)$. However, this approach may lack the expressive power to approximate the optimal policy π^* in complex scenarios.

In these cases, we may resort to DL models to parametrize the action preferences. NNs are a natural generalization of the linear parameter approach that we can tune to increase the expressive power by, e.g., increasing the number of hidden layers or their size. This way, the NN parametrizing the policy takes a state representation in the input layer $\phi(s)$, and has as many neurons as possible actions in the output layer, which encode $x(s, a)\ \forall a \in \mathcal{A}$. A softmax activation function in the output layer provides $\pi_\theta(a|s)\ \forall a \in \mathcal{A}$, as in Eq. (2.50).

The training process is analogous to training a supervised classifier on the experience gathered by the agent. Implementing REINFORCE with gradients from Eq. (2.45) is equivalent to performing gradient descent with a modified categorical cross-entropy loss (recall Eq. (2.2)):

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \gamma^t (G_{ti} - b(s_{ti})) \log \pi_{\boldsymbol{\theta}}(a_{ti}|s_{ti}), \tag{2.51}$$

where i denotes the index in a batch of n trajectories. The procedure is analogous to training a classifier in which the actions act as state labels. The main difference with supervised classification problems is that, given a state, we do not know the true probability distribution of the actions (true labels), as that would be given by the optimal policy. Instead, we assign the obtained return G_t as true label for the taken action a_t.[11] Intuitively, in classification problems we aim to enhance the probability

[11] The standard categorical cross-entropy would be $\mathcal{L} = -\frac{1}{n} \sum_n \sum_k p(a_k) \log \pi_{\boldsymbol{\theta}}(a_k|s)$, where $p(a_k)$ is the true probability distribution that we want to learn. In standard classification problems, this is typically 1 for the true label and 0 for the rest. Here, it corresponds to the optimal policy $p(a_k) = \pi^*(a_k|s)$. Since we do not have access to π^* (it is our goal!), we use the return G_t for the chosen action in its place, as π^* would favor actions with high returns. This effectively removes the expectation over actions, and we make the sum over time explicit in Eq. (2.51).

that the model provides the right label, whereas here we reinforce the actions with high returns.

In many situations, actions can take a range of continuous values rather than a discrete set of categories. For instance, a robotic arm may rotate by a certain angle or we can tune various continuous parameters in an experimental setup. Sometimes, we can discretize the action space into small intervals at the cost of a loss in precision and an increasing amount of actions, as we do in the deep REINFORCE implementation of Chapter 5. Nevertheless, this may not always be possible depending on the problem requirements and the resulting number of actions.

In these cases, we model the stochastic continuous actions with a mean μ and a standard deviation σ, such that

$$a = \mu + \sigma\xi\,, \tag{2.52}$$

where ξ is a random normal variable with unit variance. Analogously to the action preferences above, we can parametrize $\mu_{\boldsymbol{\theta}}(s), \sigma_{\boldsymbol{\theta}}(s)$ in various ways, ranging from a set of linear parameters, e.g., $\mu_{\boldsymbol{\theta}}(s) = \boldsymbol{\theta}^T\boldsymbol{\phi}(s)$, to a complex DL model with two output neurons that determine both $\mu_{\boldsymbol{\theta}}(s)$ and $\sigma_{\boldsymbol{\theta}}(s)$ for the given observation. Formally,

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sigma_{\boldsymbol{\theta}}(s)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{a-\mu_{\boldsymbol{\theta}}(s)}{\sigma_{\boldsymbol{\theta}}(s)}\right)^2\right). \tag{2.53}$$

In many cases, as the learning advances, and the agent becomes better at taking the right actions (choosing $\mu_{\boldsymbol{\theta}}(s)$), the deviations decrease and we obtain a quasi-deterministic policy.

2.3.4 Actor-critic methods

On the one hand, value-based RL methods excel at dealing with discrete state−action spaces, and their TD character makes them data efficient and allows them to tackle continuing tasks (infinite episodes). However, they experience difficulties to deal with large state−action spaces, and can't deal with their continuous version. Furthermore, they are bound to implement deterministic greedy policies, while many problems present stochastic optimal policies. Finally, small changes in the value functions can cause large variations in the policy, which may cause instabilities in learning.

On the other hand, policy-gradient methods overcome the aforementioned limitations of value-based methods, provided that they can deal with continuous (infinite) state−action spaces, and they are based on continuous stochastic policies. This ensures smooth changes in the policy throughout the learning process, and can become deterministic when needed. However, the learning happens at the end of the episodes, once we know the return, which is an issue for long trajectories or continuing tasks.

Actor-critic algorithms combine value-based and policy-based methods in order to obtain the best of both approaches. We can understand actor-critic methods as the TD version of policy gradient, with which we retain all its advantages and overcome its major limitation. It features two main elements: the *actor*, a parametrized policy that dictates the decisions, and the *critic*, a model that evaluates them.

The presence of the critic allows the agent to immediately learn from each action without waiting for the outcome at the end of the episode. Evaluating the policy mainly consists on learning its value functions, which allows the critic to assess

whether the actions are more or less favorable. In Section 2.3.3, we introduce the state-value function, $V_\pi(s)$, as the optimal baseline to reduce the variance in policy gradient. Although, in this case, we only look at $V_\pi(s)$ of the initial state in in the transitions, which does not allow us to evaluate the actions.[12]

However, we sow that, with such baseline, we can compute the gradient in terms of the advantage $A(s,a)$, defined in Eq. (2.22). The explicit form of the advantage sets the foundation for actor-critic methods [85–87]:

$$A(s_t,a_t) = \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)]\,, \tag{2.54}$$

which is derived from Eqs. (2.22) and (2.24). This expression lies at the core of TD algorithms, as it corresponds to the TD error from Eq. (2.30).

In Eq. (2.54), we use $V_\pi(s)$ to evaluate both the initial and final states of a given transition, thus constituting a critic of the action. This allows the agent to learn from every time step in REINFORCE, processing states, actions and rewards as they occur, like the TD algorithms from Section 2.3.2. Nevertheless, this advantage comes as the cost of learning two models: the policy $\pi_{\boldsymbol{\theta}}(a|s)$, and the state-value function $V_\pi(s;\boldsymbol{w})$, which are usually parametrized with NNs with parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$, respectively. The NN parametrizing the state-value function takes a feature representation of the state, $\phi(s)$, in the input layer, and has a single output neuron encoding $V_\pi(s;\boldsymbol{w})$. The policy parametrization is the same as in Section 2.3.3. We train both models simultaneously by following Algorithm 3.

Algorithm 3 Actor-critic

Require: learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{w}}$, maximum time T
Require: randomly initialized differentiable policy $\pi_{\boldsymbol{\theta}}(s|a)$
Require: randomly initialized differentiable state-value function $V_\pi(s;\boldsymbol{w})$
  while not converged **do**
    Initialize s_0
    for $t = 0$ to $T-1$ **do**
      Take action $a \sim \pi_{\boldsymbol{\theta}}(a|s)$
      Move to next state s' and obtain reward r
      $A \leftarrow r + \gamma V_\pi(s';\boldsymbol{w}) - V_\pi(s;\boldsymbol{w})$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t A \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$ ▷ Update actor
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} A \nabla_{\boldsymbol{w}} V(s;\boldsymbol{w})$ ▷ Update critic
    end for
  end while
  return $\boldsymbol{\theta}$, $\boldsymbol{w}$ ▷ Optimal actor and critic parameters

We train the actor with the methods from Section 2.3.3, and the critic using the principles from Section 2.3.2. Hence, all the methods in both sections apply to this algorithm. The parameter updates in Algorithm 3 come from performing gradient ascent with Eq. (2.45) on the actor, and an analogous update rule to Eq. (2.32) for the critic, using $V_\pi(s)$ instead of $Q_\pi(s,a)$. The process is equivalent to perform gradient descent on the losses $\mathcal{L}_{\boldsymbol{\theta}} = \frac{1}{n}\sum_n \sum_t \gamma^t A(s_t,a_t;\boldsymbol{w}) \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$, and $\mathcal{L}_{\boldsymbol{w}} = \frac{1}{n}\sum_n A(s,a;\boldsymbol{w})^2$, respectively, in which we omit the index for the sum over n samples. They are based on the same principles as the ones in Eqs. (2.35) and (2.51).

[12] In order to determine the quality of an action, we need to compare the initial and final positions. In a game, an action that escapes from the brink of a loss toward a less disadvantageous position may be more valuable than one that moves from an already favorable position to a slightly better one, despite the latter providing a higher final state-value function.

This method is often referred to as advantage actor-critic (A2C). It has been further enhanced using asynchronous actors, giving raise to the asynchronous advantage actor-critic (A3C) algorithm [88]. Other improvements rely on implementing more advanced optimization techniques, such as the natural gradient [89], as in natural policy gradient [90], natural actor-critic [91, 92], or more involved parameter updates such as trust-region [93, 94] or proximal policy optimization algorithms [95].

Part I

ML in quantum physics: quantum many-body systems and quantum technologies

Chapter 3

Machine learning in the quantum sciences

As we discuss in Chapter 1, the far-reaching influence of artificial intelligence (AI) extends across the whole scientific landscape. In this part of the thesis, we focus on the impact of machine learning (ML) in the field of quantum many-body physics. In this introductory chapter, we provide a brief review that spans both foundational aspects and practical applications, particularly in the domain of quantum computation. This sets the stage to describe our contributions to the topic in Chapters 4 and 5.

3.1 The quantum many-body problem

According to the postulates of quantum mechanics, the state of an isolated physical system is captured by a complex entity known as the *wave function* $|\psi\rangle$ that is a vector in a Hilbert space. The Born rule provides a probabilistic interpretation of the quantum state, stating that the probability to find the system in state $|s\rangle$ is precisely determined by the modulus square $|\langle s|\psi\rangle|^2$, where $\langle s|\psi\rangle$ is known as the *probability amplitude*. For instance, for a single isolated particle with two possible discrete quantum states, like a qubit with states $|0\rangle$ and $|1\rangle$, the wave function takes the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Here, $\alpha, \beta \in \mathbb{C}$ are the probability amplitudes respectively corresponding to $\alpha = \langle 0|\psi\rangle$ and $\beta = \langle 1|\psi\rangle$. Since they dictate the probability of the system being in either state, they are subject to the normalization constraint $|\alpha|^2 + |\beta|^2 = 1$. In general, we can represent the wave function as a complex-valued vector that contains the probability amplitudes for each state, and it is normalized in accordance to the L^2-norm: $||\psi||_2 = \sqrt{\sum_s |\langle s|\psi\rangle|^2} = 1$.

The Hilbert space contains all the possible states the system can be in, which is a combinatorial space that grows exponentially with the number of particles. For a system composed of N qubits, there are 2^N possible states, meaning that the wave function contains 2^N coefficients. This exponential complexity makes it extremely difficult to study quantum systems with more than a few interacting particles. Indeed, the realization that approximations would be indispensable to address any significant problem became evident early in the development of quantum theory [96], and this came to be known as the *quantum many-body problem*. In any material, there is a number of particles of the order of the Avogadro number $\sim 10^{23}$. However, a system with just about 170 qubits already has a Hilbert space of a size equivalent to the total number of atoms on Earth, around 10^{50}, and with little more than 500 qubits the many-body wave function has a staggering 10^{160} probability amplitudes, nearly the square of all the atoms in the observable universe, approximately 10^{80}.

The exploration of quantum many-body systems has unfolded along two main paths: performing extensive numerical simulations, and developing quantum simulators. On one hand, numerically simulating quantum many-body systems relies heavily on the design of smart algorithms and efficient approximations. Nowadays, quantum chemistry simulations consume a significant portion of the world's high-performance computing resources to understand the behaviour of electrons in chemical reactions, which is essential to design new drugs and materials. On the other hand, the concept of quantum simulator, originally introduced by Feynman in 1981 [97], involves engineering simple yet highly controllable quantum systems to emulate the physics of more complex systems of interest. They have proven a powerful tool to understand quantum physics, from witnessing the quantum phase transition [98], to providing a better understanding of high-temperature superconductivity [99]. Interestingly, the term "quantum simulator" is currently reserved for specialized systems capable of addressing a limited set of problems. Whereas universal quantum simulators, which are quantum systems capable of emulating any other physical system in nature, are known as quantum computers [100].

While both numerical and quantum simulation approaches have seen immense advances in the past decades, their applicability remains limited to relatively modest systems. To overcome the challenges posed by the quantum many-body problem, researchers across several domains in quantum physics are increasingly turning to machine learning (ML) in pursuit of novel techniques to deal with it. In the upcoming sections, we present an overview of some of the most noteworthy applications of ML in the field of quantum physics.

3.2 Machine learning for quantum many-body physics

ML methods are assuming an increasingly important role in the study of quantum many-body physics, finding countless applications from numerical calculations to experimental design and result analysis [37, 101, 102]. The quantum many-body problem, akin to the curse of dimensionality (recall Section 2.2.1), presents a formidable challenge that can be circumvented with dedicated ML algorithms in some cases. Furthermore, the intrinsic stochastic nature of quantum mechanics provides a rich data ecosystem where ML algorithms can thrive.

A prominent application of ML in physics is phase classification. The different phases of matter are characterized by the collective behaviour of interacting particles, whose complexity increases exponentially with the number of particles. Hence, we typically resort to defining appropriate *order parameters* [103], which describe just the relevant macroscopic degrees of freedom. However, determining these order parameters for novel phases of matter is an exceptionally challenging task. ML algorithms have emerged as an alternative approach to identify phases and phase transitions both in simulated [104–109] and experimental data [110–112]. Leveraging their proficiency in classification tasks afflicted by the curse of dimensionality, such as image classification [43], ML algorithms are invaluable at unraveling the tapestry of phases of matter.

In numerical simulations of quantum many-body systems, we need efficient representations of the wave function that do not take exponentially-many resources. Often, the physically-relevant states are confined within a low-dimensional corner of the Hilbert space, which we attempt to represent with parametrized wave functions. Neural networks (NNs) emerge as particularly well-suited candidates for this task, being universal approximators, which gives raise to the name: neural

quantum states (NQSs). Since their introduction by Carleo and Troyer [113], multiple NN architectures have been explored [114–118], and it has been shown that they can efficiently encode highly entangled quantum states [119, 120]. Notably, the quantum states representable by NNs are a superset of those captured by tensor networks [121] – an alternative efficient approach to represent quantum many-body wave functions [122]. NQSs have found considerable success in the study of highly-correlated systems in condensed matter [123–125], and addressing challenges in quantum chemistry [126], with notable models such as the *FermiNet* [127, 128] and the *PauliNet* [129]. We refer to our hands-on tutorial, accessible in Ref. [130], for an introduction to NQSs.

A direct consequence of the introduction of NQSs has been a remarkable acceleration of Monte Carlo simulations. Traditional Monte Carlo simulations rely on iteratively sampling states by performing local modifications to the previous ones [131], which results in highly-correlated samples. Prior efforts to mitigate these correlations involved ML-based approaches to guide the sampling process [132, 133], and some even imported ideas from reinforcement learning (RL) [134, 135]. However, the introduction of generative models as NQSs has solved the correlation problem by allowing us to draw uncorrelated samples directly from the model in parallel, a concept that has transcended quantum physics [116, 117, 136, 137]. These advanced sampling schemes have also been seamlessly integrated into classical optimization processes, such as simulated annealing, to tackle long-standing challenges like the study of spin-glasses [138]. Beyond the improved sampling, theoretical advancements in different ML areas, such as RL [139], are reshaping quantum Monte Carlo algorithms from the foundations leading to faster and more stable simulations [140]. Furthermore, ML tools not only accelerate, but also enhance the accuracy of simulations, as exemplified in molecular dynamics simulations [10, 141, 142].

Exploring new experiments is crucial for the development of science. However, quantum experiments pose daunting technical and scientific challenges, spanning from conceptualization, to execution, and subsequent analysis. As the pursuit of knowlege leads to increasingly complex challenges, the necessary experimental setups also become equally intricate. The design of novel experiments involves the combination of advanced knowledge with a thorough exploration of the parameter spaces of the experiments [143]. AI pipelines excel at this task proposing new experimental setups based on iterative feedback [144], with most applications focusing on quantum optics [22, 145–147]. Furthermore, they also prove invaluable tools to control the experimental parameters both in offline [148–150] and online [151–153], facilitating the development of advanced protocols such as rapid cooling schemes [154, 155]. Given the substantial data output of quantum experiments, ML algorithms emerge among the best tools for the result analysis. This entails not only drawing meaningful conclusions, but also certifying the experiment. When evaluating the experiment's output, a natural question may arise: is it behaving as intended? We can attempt to answer it through various means depending on the application, for instance, performing quantum state tomography [156–158] or Hamiltonian learning [159–161], which consists on reconstructing the effective Hamiltonian from measurements. Finally, ML aids in drawing conclusions from the results, for example, clarifying the choice between competing theories [110].

In summary, ML assumes a pivotal role across the entire spectrum of quantum physics research: from purely theoretical applications, to contributing to all the stages of experimental research. In the upcoming section, we dive into specific applications of ML in the development of quantum computers and, in Chapter 4, we detail our contribution with ML to fundamental quantum information research.

3.3 Machine learning for quantum computing

With the development of novel technologies, there come a myriad of challenges across multiple levels. ML techniques catalyze the development of quantum computation in all possible facets, from the construction and control of quantum computers, to the design and implementation of quantum algorithms [37, 101, 102].

Nowadays, gate-based quantum circuits are the predominating language to describe quantum computations [162]. This framework describes the evolution of a set of qubits through a series of discrete operations known as *quantum gates* in a hardware-agnostic fashion. Quantum gates are the quantum analog to the logic gates used in classical computing, and they are represented by unitary matrices acting on a set of qubits. Quantum algorithms are constituted by the conjunction of gates, their parameters, and the qubits they act upon. Some renowned examples include the quantum Fourier transform [163], upon which Shor's factorization algorithm builds upon [164], Grover's search algorithm [165], or the HHL algorithm to solve systems of linear equations [166]. However, these algorithms demand a substantial amount of gates and qubits for any significant application, rendering them impractical for the current noisy intermediate-scale quantum (NISQ) computers [167].

While NISQ devices are constrained to small and shallow circuits, they already serve as valuable testing grounds for new ideas. A notable application for NISQ devices is in variational quantum algorithms, such as the variational quantum eigensolver [168] or the quantum approximate optimization algorithm [169]. These algorithms operate in a quantum-classical hybrid optimization scheme: first, a parametrized quantum circuit is used to evaluate a cost function and, subsequently, a classical optimization scheme updates the circuit's parameters based on the cost, analogous to the prototypical training loop in ML. Nevertheless, these algorithms suffer from major shortcomings. Notably, obtaining gradients from the circuit parameters proves challenging, provided that existing approaches scale poorly with the number of parameters [170], and the optimization schemes struggle with exponentially-vanishing gradients, referred to as *barren plateaus* [171]. Ongoing efforts aim to develop more efficient methods to extract gradients [172], and to better comprehend and mitigate the optimization issues, drawing inspiration from techniques originally designed to train and interpret ML models [173, 174]. Furthermore, ML algorithms can be applied directly to perform the parameter optimization, for example, framing it as an RL problem [175], or even to find a suitable parameter initialization of the circuit [176].

ML methods find multiple applications in the development and enhancement of quantum algorithms. For instance, state preparation, a recurrent task in quantum computing, can be naturally framed as an RL problem [22, 149, 177, 178]. Even more, ML algorithms contribute to the design of novel quantum algorithms and subroutines [179, 180], as well as the reduction of the total number of gates in quantum circuits [181], which is critical for NISQ computers. However, many of these applications rely on quantum circuit simulators, such as those provided by quantum computing libraries like PennyLane [182] or Qiskit [183]. As we have introduced earlier, simulating quantum computers is inherently challenging (we would not need them otherwise!), limiting these applications to just a few qubits. To address these challenges, ML methods offer tools to simulate quantum circuits [184, 185], similar to the role they play simulating quantum many-body systems, and provide alternative approaches to circumvent the simulation requirements, as seen in circuit synthesis with generative models [186].

Ultimately, quantum algorithms are meant to run in a quantum computer, presenting additional challenges. Various quantum computing platforms are based on different qubit technologies, such as superconducting circuits [187], photonic platforms [188, 189], trapped ions [190], or Rydberg atoms in optical tweezers [191, 192]. Each platform provides different interactions and controls that determine the *native gates* of the device, which are the operations that can be naturally executed in the hardware. Consequently the originally hardware-agnostic quantum circuits must undergo a compilation process, rewritting them in terms of the hardware's native gates. The compilation of quantum circuits is based on the Solovay-Kitaev theorem, which asserts the existence of multiple equivalent representations of quantum circuits using different sets of gates while yielding the same output [193, 194]. Nevertheless, decomposing a circuit in terms of a different set of gates can be very challenging. Typically, there exists a trade-off between compilation and execution time, which can be circumvented by framing the compilation process as an RL problem [195], for instance. We provide a practical introduction to circuit compilation in our tutorial, accessible in Ref. [196], where we show how to repeatedly perform subtle transformations to quantum circuits to reduce the amount of gates, and rewrite them in terms of different gate sets.

Another significant challenge that comes with the quantum hardware is preserving the quantum information and ensuring that gates are applied flawlessly. Both quantum and classical computations are susceptible to errors, requiring schemes to either prevent them or mitigate their impact. Classical error correction schemes, however, cannot be directly applied to quantum computing due to the no-cloning theorem [197], which forbids copying quantum states, and the fact that directly measuring them to identify errors erases the superposition. Most quantum error correcting schemes employ quantum codes to encode a logical qubit in the state of multiple physical qubits [198–200]. Specific measurements can reveal error *syndromes*, indicating where errors have occurred, although syndromes themselves are prone to errors too. Correcting errors based on imperfect syndrome information can be framed as an RL problem [201, 202], leading to the first-ever error-corrected logical qubit in a superconducting quantum computer [203]. Additionally, other RL-based error-correcting schemes focus on finding algorithmic strategies and subroutines to preserve multi-qubit states [204], or even to design quantum codes [205]. In parallel to error correction, error suppression strategies aim to minimize errors rather than directly fixing them. For instance, Hamiltonian learning is employed to assess whether error correcting codes are implemented correctly [206]. In a broader context, RL algorithms are used to optimize gate implementations at the hardware level, considering the specific noise characteristics of the devices [207, 208]. Overall, ML techniques are integral in nearly every stage of error suppression, even contributing to the design and optimization of entirely automatic error-suppression pipelines. These pipelines may involve error-aware quantum circuit compilation, system-wide gate optimization, circuit-level error cancellation, and measurement error mitigation, among other strategies [209].

Finally, despite our primary focus lies on the contributions of ML to the development of quantum computing, there exists a noteworthy research field dedicated to enhancing ML algorithms with principles from quantum physics [210, 211]. Although the quantum ML field holds promises for computational advantages [212–214], it is still in its early stages. A significant research avenue focuses on using parametrized quantum circuits to evaluate compute-intensive parts of ML algorithms,

for instance in kernel methods [215, 216] or RL [217, 218]. Concurrently, there are extensive efforts to characterize quantum ML algorithms at a fundamental level identifying their strengths and weaknesses [219–222].

In conclusion, ML is essential for the development of new technologies such as quantum computing at all levels: from the design of quantum algorithms, to the simulation and optimization of the hardware. In Chapter 5, we detail our implementation in PennyLane [182] of an experimentally-friendly gate calibration scheme based on RL.

Chapter 4

Certificates of many-body quantum physics assisted by machine learning

In this chapter, we describe a reinforcement learning (RL) method designed to aid in the exploration of quantum many-body systems. The approach offers a systematic framework for the construction of relaxations of complex problems, finding applications across multiple fields. Especially in the quantum information processing field, where relaxation methods are at the forefront of research.

The content of this chapter is based on the work we presented in Ref. [223]. Accompanying this work is a Python library, accessible in Ref. [224], containing the code with comprehensive documentation and tutorials to reproduce the presented results.

4.1 Approximate methods to study quantum physics

As we discuss in Chapter 3, we frequently encounter computationally intractable tasks in the exploration of quantum physics, often related to the quantum many-body problem. To overcome these challenges, we usually rely on methods that offer accurate approximations to the actual solutions. There exist two main paradigmatic approaches: the variational ansatz and relaxation methods, illustrated in Fig. 4.1.

On the one hand, the variational ansatz involves parametrizing a family of solutions with the hope that it contains a good approximation to the optimal one, depicted in Fig. 4.1(a). This versatile approach has proven remarkable success across multiple domains in the quantum sciences, for instance, in quantum chemistry [168, 225–227], condensed matter [113, 122, 131, 228–230], and quantum machine learning [210, 211]. Furthermore, it is the foundational pillar upon which many modern quantum algorithms rest [168, 169, 187, 230–234]. Indeed, in the previous Chapter 3, we describe some prominent examples of variational methods in combination with machine learning (ML) techniques to study quantum systems, such as neural quantum states (NQSs). Variational approaches yield an upper bound (in a minimization problem) to the optimal solution, although the result strongly depends on the suitability of the ansatz for the particular task. Increasing the complexity of the ansatz may yield better approximations, although at the expense of increasing the overall computational cost. Additionally, the distance between the obtained solution and the optimal one is unknown, in general. Even in the cases when we obtain it, we need additional methods to prove it is indeed the case.

On the other hand, we find relaxation methods. Any optimization task is characterized by the problem's constraints. All the points that fulfill them define the

feasible set, which may be hard to optimize over in many cases. Therefore, to ease the optimization process, we may consider relaxing some of the constraints, obtaining a relaxed larger set, as illustrated in Fig. 4.1(a). However, the minimum over a larger set can only be less or equal than the original one, thus resulting in a lower bound to the problem. Hence, the combination of the two approaches, variational and relaxations, yields an upper and lower bound that conform an uncertainty interval around the optimal solution, as illustrated in Fig. 4.1(b).

Relaxation techniques are widely used in quantum information processing. In this field, semidefinite programming (SdP) has been a successful and recurrent tool to build relaxations [235–237]. Perhaps, the most paradigmatic example in the context of entanglement theory is the Peres criterion, which is a relaxation from the set of separable states to the set of states that are positive under partial transposition (PPT) [238]. The membership problem in the separable set is NP-hard [239], whereas checking the PPT criterion is very simple, yielding one of the simplest ways to show that a quantum state is entangled. However, not all quantum states in the PPT set are separable, as the relaxed set contains states that are both entangled and PPT [240]. Relaxation techniques also play a major role in the device-independent version of quantum information processing [241]. For instance, in cryptographic security proofs, we need to consider all possible quantum attacks, which are hard to characterize, motivating research for supraquantum theories that are analytically tractable [242, 243]. In the quest to characterize the set of quantum correlations [244], several operationally simple, outer approximations have been proposed [245–252], as well as systematic relaxations via SdP [236, 253, 254]. Additionally, relaxation methods have found a wide range of applications in quantum physics and chemistry [255–261]. A recurrent theme is to find solutions that are simple enough to be understandable and computationally tractable, while being as accurate as possible.

Strongly relaxed problems may be easier to solve, although they may come at the expense of providing looser bounds. Simultaneously, the effectiveness of relaxations may vary, with some featuring a more elegant and smarter design than others, yielding better bounds while using similar computational resources. The success of this approach generally relies on the exploitation of useful properties of the system, such as the existence of symmetries. However, without any prior knowledge about the problem at hand, or whenever it lacks such appealing properties, efficient relaxations may be highly elusive. Hence, it is crucial to devise methods to find, among all possible relaxations of the original problem, the best trade-off between accuracy and simplicity. Finding such optimal relaxation is a complex combinatorial optimization problem whose solution can reveal relevant properties of the underlying system.

In Ref. [223], we propose to harness the power of machine learning (ML) to systematically obtain optimal relaxations. ML techniques have shown great success at dealing with combinatorial problems [262], and there are several valid approaches, from the supervised learning of neural networks (NNs) [263] to unsupervised methods over graphs [264] and, in particular, RL [265]. While traditional algorithms rely on heuristics and specific insight about the problem, ML approaches are able to solve many of them faster and without any prior knowledge or assumption. We combine RL techniques with SdP to systematically search for optimal relaxations within a finite computational budget.

In the proposed scheme, an RL agent has access to a black box that computes the

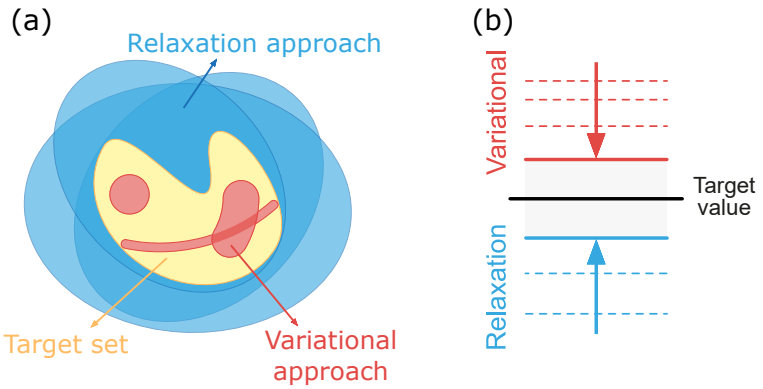

FIGURE 4.1: **Interpretation of exact solutions and bounds obtained through variational and relaxation methods.** (a) Schematic representation of an optimization task. The goal is to optimize a function over a hard to characterize set (yellow set). The variational approach allows us to parametrize subsets within the one of interest (different red sets). Different parametrizations yield different subsets that are more or less convenient depending on the task. Relaxation techniques can efficiently represent larger sets than the one of interest (different blue sets) exploiting, for instance, convexity or linearity. Neither different variational approaches nor different relaxations need to be contained into one another, so the sets they represent are, in general, incomparable. (b) Values of the objective function. In black, the optimal unknown value. In red, the different minima obtained by variational methods. The smaller the value, the better the bound. In blue, different minima obtained by relaxation techniques. The greater the value, the more accurate their associated certificate. In grey, the uncertainty region where the optimal solution lies, given by the best variational and the best certificate obtained so far.

relaxation of the problem by solving an SdP[1] (see Fig. 4.3(a)). The agent can increase or decrease the relaxation level and observe an output that depends on both the associated computational cost and the quality of the obtained bound. Finding useful relaxations can be seen as a meta-optimization with a wide range of applicability. Here, we present it in two paradigmatic problems in quantum physics and quantum information: finding the ground state energy of local many-body Hamiltonians and building energy-based entanglement witnesses.

4.2 Relaxations with semidefinite programming

In this section, we introduce the methods to systematically build relaxations with SdP. These relaxations are based on the construction of an SdP, whose optimal solution constitutes a certificate, as we detail in the following Section 4.2.1. In the interest of simplicity, we introduce the main concepts applied to the optimization tasks considered throughout the chapter: finding the ground state energy of quantum local Hamiltonians and building energy-based entanglement witnesses. The mathematical formalism of the second task is an extension of the first one. Hence, we show here how to relax the ground state energy problem and, then, extend it to the entanglement witnessing in Section 4.6.1.

[1]We slightly abuse notation and let SdP denote both "semidefinite programming" and a "semidefinite program". It is always clear from the context.

4.2.1 The problem of ground state energy approximation

Consider the optimization task to find the ground state energy, E_0, of a quantum local Hamiltonian

$$H = \sum_{i=1}^{m} H_i \,. \tag{4.1}$$

The Hamiltonian H acts on n qubits, and it is a sum of terms H_i, each of which acts on at most $k = O(1)$ qubits. The sum Eq. (4.1) has therefore $m = O(\text{poly}(n))$ terms. The support of H_i, denoted $\text{supp}(H_i)$ is the set of qubits where H_i acts non-trivially. The supports of the different H_i may overlap, such that $\text{supp}(H_i) \cap \text{supp}(H_j)$ may not be empty.

To find E_0, a possibility is to directly construct a quantum state that has E_0 energy with respect to H. Therefore, a first possible approach is to parametrize a family of quantum states $|\psi(\boldsymbol{\theta})\rangle$ exploiting some known properties of H. We can safely assume the parametrization yields a valid (normalized) quantum state for any value of the parameters $\boldsymbol{\theta}$. Additionally, by construction, $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle \geq E_0$ for all $\boldsymbol{\theta}$. Let us denote

$$\gamma = \min_{\boldsymbol{\theta}} \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle, \tag{4.2}$$

which satisfies $\gamma \geq E_0$ by construction. An example of such a parametrization would be to describe $|\psi(\boldsymbol{\theta})\rangle$ as a tensor network contraction, which exploits the locality properties of H, limiting the entanglement present in its ground state [122, 266, 267].

Complexity theory results (in particular, QMA-hardness) strongly suggest that finding, or even approximating, the ground state energy of a local Hamiltonian is a hard task, even for a quantum computer [268–270]. Furthermore, this hardness persists in physically relevant instances [271]. Notice that, even if we happen to find the actual optimal solution $|\psi(\boldsymbol{\theta})\rangle$, we cannot prove that it is the global minimum solely from that [272].

It is therefore highly desirable to obtain a bound β from the other side for which $E_0 \geq \beta$ can be proven. This guarantees $E_0 \in [\beta, \gamma]$, as illustrated in Fig. 4.1(b), thus, it can help determine whether it is worth to refine the search depending on $|\gamma - \beta| < \varepsilon$. However, for a proof of the type $E_0 \geq \beta$, constructing an example $|\psi(\boldsymbol{\theta})\rangle$ is not good enough. We need a proof that is satisfied by all valid quantum states and, possibly, a larger set, as long as it makes the proof simpler. Such a proof is referred to as a certificate, and it is typically obtained by numerical means. In the upcoming Sections 4.2.2 and 4.2.3, we show how SdP is a natural tool to obtain such certificates and how to systematically improve them.

4.2.2 Building a trivial relaxation

A common technique to build relaxations for the local Hamiltonian problem is via the triangle inequality [255, 273–275]:

$$\min_{\rho} \text{Tr}[\rho H] \geq \sum_i \min_{\rho_i} \text{Tr}[\rho_i \hat{H}_i] \,, \tag{4.3}$$

where ρ and ρ_i are density matrices acting on the support of H and H_i respectively. Note that i refers to a Hamiltonian term and it has nothing to do with the i-th party. Furthermore, in Eq. (4.3), the $\hat{H}_i$ are sums of some local terms H_j of Eq. (4.1), grouped so that $\text{supp}(\hat{H}_i)$ is as large as possible while still allowing for computation of their minimal eigenvalue. This size is directly related to the available computational resources.

Let us observe that the right-hand side in Eq. (4.3) is a sum of minima, where each minimization is carried out independently. Due to this independence, in general, it is not the case that different ρ_i are mutually compatible, i.e., that there exists a global state ρ such that each ρ_i is the corresponding partial trace of ρ. The converse is true, however: every valid quantum state ρ has an associated set of partial traces ρ_i, but given a set of ρ_i, a global ρ may not exist. This is what proves the inequality Eq. (4.3).

The minimization of the right-hand side of Eq. (4.3) is equivalent to solving the following SdP:

$$\begin{aligned} \beta_\varnothing := \quad & \min_{\{\rho_i\}} \quad \textstyle\sum_i \mathrm{Tr}[\rho_i \hat{H}_i] \\ & \text{s.t.} \qquad\quad \rho_i \succeq 0 \\ & \qquad\qquad\quad \mathrm{Tr}[\rho_i] = 1\,. \end{aligned} \tag{4.4}$$

Since there is no mutual compatibility enforced among the ρ_i, and each one is treated independently, the triangle inequality Eq. (4.3) constitutes a trivial relaxation.

4.2.3 Building tighter relaxations

A natural way to strengthen the relaxation resulting from the triangle inequality is to impose further restrictions on the collection of possible ρ_i, in such a way that any quantum state would also satisfy them. The strongest restriction possible is to directly ask that $\{\rho_i\}$ come from a global quantum state. Unfortunately, this is equivalent to finding the value of E_0, which is QMA-complete. Furthermore, it is strongly connected to solving the so-called quantum marginal problem (QMP), which is also QMA-complete [268–270]. The QMP has been solved completely in very rare instances, such as the global state being symmetric [276] or for the case of one-body marginals [277–279] Nevertheless, the SdP-based formulation Eq. (4.4) motivates a hierarchy of relaxations based on solving the QMP up to some degree of compatibility.

It would be natural to expect that, at least, the partial traces where the supports from $\{\rho_i\}$ intersect match between them. This reduces the space of solutions, provided that $\{\rho_i\}$ must fulfill additional conditions. Since the minimization is over a smaller set, its result can only be a tighter bound.

Hence, the first level of compatibility we might want to ask for is that ρ_i and ρ_j yield the same reduced density matrix (RDM) on their common support, which we shall denote $\rho_{i\wedge j}$:

$$\mathrm{Tr}_{\mathrm{supp}(\rho_j)^c}[\rho_i] = \mathrm{Tr}_{\mathrm{supp}(\rho_i)^c}[\rho_j] \equiv \rho_{i\wedge j}\,. \tag{4.5}$$

Here, the partial trace $\mathrm{Tr}_S(\cdot)$ denotes that we eliminate subsystem S and the superindex c indicates the complementary set. Thus, Tr_{S^c} produces the RDM acting on the subsystem S. Note that the partial trace condition is linear in ρ_i. Therefore, it can be naturally imported into Eq. (4.4) and still be formulated in terms of an SdP:

$$\begin{aligned} \beta_1 := \quad & \min_{\{\rho_i\}} \quad\quad\quad\quad\quad\quad \textstyle\sum_i \mathrm{Tr}[\rho_i \hat{H}_i] \\ & \text{s.t.} \qquad\quad\quad\quad\quad\quad\quad\quad \rho_i \succeq 0 \\ & \qquad\quad\quad\quad\quad\quad\quad\quad\quad\quad \mathrm{Tr}[\rho_i] = 1 \\ & \qquad\quad\quad\quad\quad\quad\quad\quad\quad\quad \mathrm{Tr}_{\mathrm{supp}(\rho_j)^c}[\rho_i] = \rho_{i\wedge j}\,. \end{aligned} \tag{4.6}$$

Given that the sets of $\{\rho_i\}$ that satisfy the constraints of Eq. (4.6) also satisfy the constraints of Eq. (4.4), we have $\beta_\varnothing \leq \beta_1 \leq E_0$, by construction.

The certificates obtained from Eq. (4.6) can be further strengthened by adding virtual RDMs. For instance, even if H is $2-$local, we might want to ask, for instance, that the two-body RDMs acting on $Alice - Bob$ and $Bob - Charlie$ are such that they

both come from a virtual three-body density matrix acting on $Alice - Bob - Charlie$. The latter is not strictly necessary in order to compute the energy, for 2−body density matrices suffice, but this compatibility condition further restricts the set $\{\rho_i\}$, therefore improving the bound. In mathematical jargon, this method is known as representing the feasible set as a projected spectrahedra [280]. Hence, instead of solely asking that ρ_i and ρ_j yield the same RDM on their intersection, now we might impose a stronger constraint, which is that ρ_i and ρ_j come from a valid density matrix $\rho_{i\vee j}$ defined on the union of their supports:

$$\begin{aligned} \beta_2 := \quad & \min_{\{\rho_{i\vee j}\}} && \textstyle\sum_i \mathrm{Tr}[\rho_i \hat{H}_i] \\ & \text{s.t.} && \rho_{i\vee j} \succeq 0 \\ & && \mathrm{Tr}[\rho_{i\vee j}] = 1 \\ & && \mathrm{Tr}_{\mathrm{supp}(\rho_i)^c}[\rho_{i\vee j}] = \rho_i. \end{aligned} \tag{4.7}$$

Note that the constraints imposed in Eq. (4.7) automatically imply those of Eq. (4.6), so we can omit them, as they became redundant.

We also observe that, although now we have $\beta_\varnothing \leq \beta_1 \leq \beta_2 \leq E_0$, the cost of solving Eq. (4.7) is substantially higher than that of Eq. (4.6), because the SdP variables $\rho_{i\vee j}$ act on more qubits than ρ_i and the cost of representing them grows exponentially in the number of qubits. Similarly, the relaxations from Eq. (4.7) can be further strengthened by considering compatibility with more regions, yielding a chain of inequalities $\beta_\varnothing \leq \beta_1 \leq \beta_2 \leq \ldots \leq E_0$.

In Eq. (4.7), the compatibility constraints are enforced on all possible pairs (i, j). However, not all the constraints are equally useful. In an extreme case, when $\mathrm{supp}(\rho_i) \cap \mathrm{supp}(\rho_j) = \varnothing$, adding the variable $\rho_{i\vee j}$ with its respective constraints makes no difference. Indeed, since $\mathrm{Tr}[\rho_i \hat{H}_i + \rho_j \hat{H}_j] = \mathrm{Tr}[(\rho_i \otimes \rho_j)(\hat{H}_i \otimes \mathbb{1}_j + \mathbb{1}_i \otimes \hat{H}_j)]$, the choice $\rho_{i\vee j} = \rho_i \otimes \rho_j$ is always possible, as it satisfies the rest of constraints, therefore not changing β_2. We remark this tensor product choice is possible because the supports do not intersect. However, if we define $\rho_{i\vee j}$ as a variable in Eq. (4.7), we increase its computational complexity without improving the bound, thus yielding a worse certificate.

4.3 The constraint space

In Section 4.2, we have seen how to build relaxations with SdP, and how to strengthen them by introducing additional constraints to obtain tighter bounds to the optimal solution of the original optimization problem. However, as we have shown at the end of the previous section, not all the constraints have the same impact in the final result, with some even exclusively contributing to increase the overall computational cost without improving the resulting bound. In this section, we present the constraint space, which is fundamental for the consistent exploration of problem relaxations, as we explain in the following Section 4.4.

Continuing with the example of finding the ground state energy of local Hamiltonians, let us consider a set of n qubits, labelled from 0 to $n-1$, and denote $[n] = \{0, \ldots, n-1\}$. Let $\mathcal{P}([n]) = \{\varnothing, \{0\}, \{1\}, \ldots, \{n-1\}, \{0,1\}, \{0,2\}, \ldots, [n]\}$ denote the parts of $[n]$, which is the set of all subsets of $[n]$, thus containing 2^n elements.

We can associate a certificate to every subset $C \subseteq \mathcal{P}([n])$ in the following way: for each element $S \in C$, corresponding to a subset of $[n]$, we consider the RDM acting on the qubits labelled by the elements in S, which we denote ρ_S. Let us denote

$\Xi_C := \{\rho_S\}_{S\in C}$ the collection of RDMs associated to C. By enforcing compatibility on their overlapping supports, we can define the SdP

$$\begin{aligned} \beta_C := \quad & \min_{\Xi_C} && \textstyle\sum_i \langle H_i\rangle && \\ & \text{s.t.} && \rho_S \succeq 0 && \forall S \in C \\ & && \mathrm{Tr}[\rho_S] = 1 && \\ & && \mathrm{Tr}_{R^c}[\rho_S] = \mathrm{Tr}_{R^c}[\rho_{S'}] \quad && \forall R \subseteq S\cap S', \quad S, S' \in C\,, \end{aligned} \tag{4.8}$$

where the partial trace over the whole system is set to one by convention $\mathrm{Tr}_{[n]}[\rho] = 1$. We have written the objective function as $\sum_i \langle H_i\rangle$ for the following reasons: first, C can be small enough so that there is no $S \in C$ such that $\mathrm{supp}(H_i) \subseteq S$. If this is the case, then we substitute $\langle H_i\rangle$ by the minimal eigenvalue of H_i, in the same spirit as the trivial relaxation from Eq. (4.4). Hence, if $C = \emptyset$, the cost function of Eq. (4.8) amounts to the sum of the minimal eigenvalue of each H_i. Otherwise, if Ξ_C contains a density matrix ρ_i whose support contains the support of H_i, we simply compute $\langle H_i\rangle = \mathrm{Tr}[\rho_i H_i]$. Note that, in case that multiple density matrices from Ξ_C could be used to compute $\langle H_i\rangle$, the last constraint of Eq. (4.8) guarantees the result is well-defined and independent of the choice $\rho_i \in \Xi_C$. In practice, the last constraint of Eq. (4.8) rarely needs to be imposed over all the subsets of the intersection, and it is enough to take $R = S \cap S'$ for all pairs $S, S' \in C$. Regardless of this constraint enforcement, the SdP yields a valid lower bound.

Furthermore, given a set of constraints $C \subseteq \mathcal{P}([n])$, it is not necessary to define Eq. (4.8) over all the variables contained in Ξ_C. If some $S \in C$ is contained in another $S' \in C$, such that $S \subseteq S'$, we can simply use $\rho_{S'}$, as it contains all the information on ρ_S. This choice is well-defined due to the constraints in Eq. (4.8) and it naturally defines a simplification function $s : \Xi_C \mapsto s(\Xi_C)$, which allows to simplify the SdP by removing redundant variables.

A fundamental aspect in the optimization of relaxations is to account for a finite computational budget, which effectively limits which constraints can be considered. The asymptotic complexity of an SdP with m variables of matrix size k depends on the method that it is used to solve it. A rough estimate is $O(m^2k^2)$, without factoring in the iteration costs of the algorithm [281]. There exist a whole plethora of algorithms displaying different complexities, such as *SeDuMi* with $\tilde{O}(m^2k^{5/2} + k^{7/2})$ for large instances and $\tilde{O}(km^3)$ with optimized algorithms for small matrices [282], or interior point methods running in $\tilde{O}(\sqrt{m}(m + k^3))$ [283]. The $\tilde{O}$ notation is used to supress $\mathrm{polylog}(mn/\varepsilon)$ terms, where ε is the required precision [284, 285]. Interestingly, quantum algorithms have been proposed to solve SdPs [286], and ML methods have been studied to aid SdP solvers [287]. In light of the whole zoo of algorithms for SdP and their various complexities, compute time is not a consistent metric to bound the constraint space. Instead, given computational budget, we determine the set of maximal (m, k) that are allowed by effectively limiting the size and contents of Ξ (see Section 4.5.2).

The space of constraints forms a partially ordered set with respect to the following partial order relation. Given $C, C' \in \mathcal{P}([n])$, we say $C \preccurlyeq C'$ if, and only if, for each $S \in C$ there exists a $S' \in C'$ such that $S \subseteq S'$. The motivation of the partial order relation $\preccurlyeq$ is that $C \preccurlyeq C'$ implies $\beta_C \leq \beta_{C'}$ by construction: every density matrix in Ξ_C can be obtained by tracing out some elements of another density matrix in $\Xi_{C'}$, and the constraints in Eq. (4.8) enforce mutual compatibility among all the elements in Ξ_C and $\Xi_{C'}$. In Fig. 4.2 we illustrate such structure, which motivates the definition of the RL formalism in the upcoming Section 4.4.

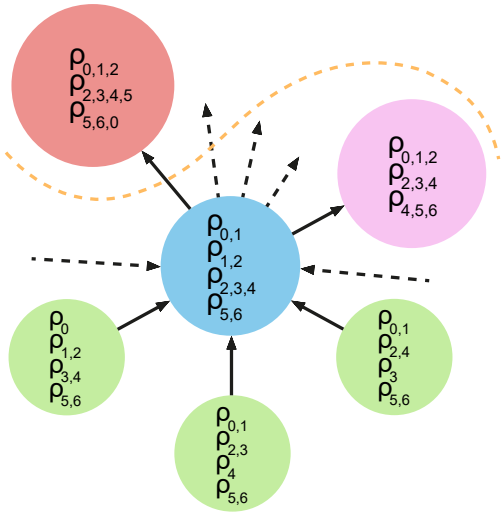

FIGURE 4.2: **Poset structure of the constraint space.** The different circles represent Ξ_C for different $C \subseteq \mathcal{P}([n])$. The arrows represent the partial order relation $\preccurlyeq$, so that $\Xi_C \preccurlyeq \Xi_{C'}$ is represented by an arrow from Ξ_C to $\Xi_{C'}$. Only the arrows relative to the central node are drawn. Dashed arrows indicate that there exist many more $\Xi_{C''}$ arriving/departing from the central node that are simply not drawn. The orange dashed line separates those Ξ_C that fall into the allowed computational budget (green, blue and pink nodes) from those that are too costly (red). Moving vertically up into the diagram provides better certificates, but at a higher cost. Since $\preccurlyeq$ is a partial order relation, some nodes (e.g., the three at the bottom) are incomparable.

4.4 Constraint optimization with reinforcement learning

In this section, we propose a method to obtain the best possible certificate within a certain computational cost by exploring the constraint space described in the previous Section 4.3. Due to the high amount of structure in this extensive combinatorial space, we propose to use reinforcement learning (RL), introduced in Section 2.3, with function approximation. In our proposed framework, it naturally favors lower cost solutions and can optimize the exploration strategy based on previous experiences. In such spaces, experience in one region may be useful in others. For instance, in periodic systems, actions in one domain should be identical to actions in another, which further allows for easy transfer of learning without explicit analysis of the model parameters (see Section 4.5.3).

To this end, we frame the optimization problem as a Markov decision process (MDP). The MDP is defined through a state space, an action space, a transition function between states given an action, and a reward function, which associates a value to each state-action-state tuple. We detail all these elements below. A learning agent, explores the constraint space with the goal to find the set of constraints $C^* \subseteq \mathcal{P}([n])$ that provides the best possible certificate within a limited computational budget, while using the least amount of resources. In algorithmic terms, we distinguish two main independent parts:

i. An environment that hosts the constraint space restricted by the computational budget, as in Fig. 4.2. It also contains a black box that takes a set of constraints C as input, computes β_C by solving the associated SdP (Eq. (4.8)) and outputs a reward.

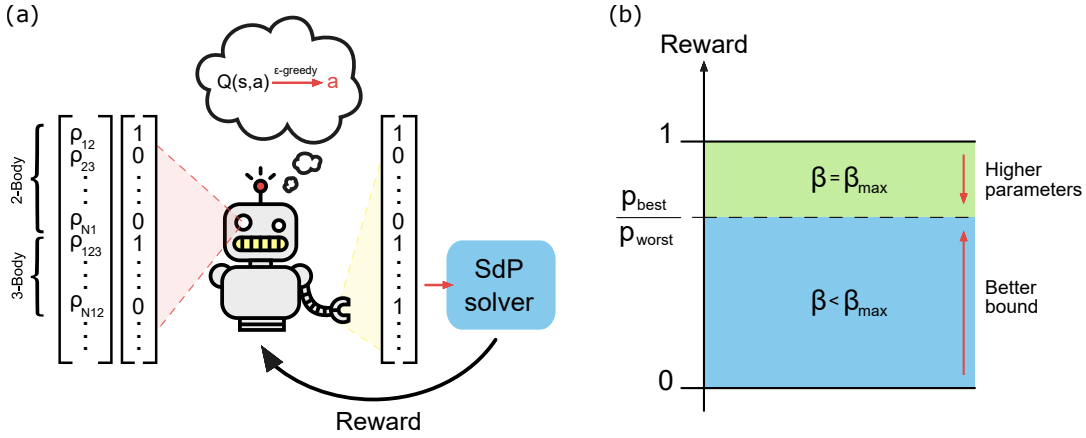

FIGURE 4.3: **Schematic representation of the RL pipeline.** (a) Representation of the reinforcement learning framework. First, the agent observes the state: a one-hot encoding of the active constraints. Given the observation, it estimates the Q-values associated to the possible actions with a deep Q-network (DQN). Then, it decides which action to take according to an ϵ-greedy policy, bringing the agent to a new state. Finally, the black box solves the SdP associated to the new state, providing the agent with a reward. (b) Illustration of the reward components from Eq. (4.9).

ii. A learning agent that navigates the environment's constraint space (i). At every point in the space, it inputs the corresponding set of constraints C into the black box. The agent can choose to strengthen or loosen the constraints, effectively exploring the constraint space with its actions. In doing so, the agent obtains different rewards that guide it towards finding the optimal relaxation. Note that the agent is completely agnostic about the actual physical problem at hand.

We aim to understand up to which extent such a fully automated approach may help in studying physical systems. In the following, we connect the MDP components to our running example. See Fig. 4.3(a) for a schematic depiction.

State space – The state space corresponds to the constraint space introduced in Section 4.3, in which each state is a set of constraints $C \subseteq \mathcal{P}([n])$ and it is bound by the computational budget, as illustrated in Fig. 4.2. We represent the states by one-hot encoding of the active constraints $S \in C$: considering a set of 2^n-dimensional canonical vectors with only a non-zero unit element, each representing an element $S \in \mathcal{P}([n])$, a state vector is the sum of the vectors that encode the components $S \in C$. Equivalently, it identifies the set $\Xi_C = \{\rho_S\}_{S \in C}$ of RDMs that enter as variables in Eq. (4.8). As shown in the leftmost part of Fig. 4.3(a), the RDMs ρ_S are ordered according to their dimension in the state vector. Out of the 2^n possible variables, we need only consider $\text{poly}(n)$ of them, effectively reducing the state vector size: we can ignore the 1-body constraints as well as those ρ_S whose sole contribution to the cost of solving the associated SdP would exceed the computational budget. With a computational budget B, this leaves $n^{O(\log(B))}$ available RDMs to construct the certificate. If no $S \in C$ is such that $i \in S$, the 1-body constraint corresponding to $\rho_{\{i\}}$ is added by default. Therefore, the smallest set of constraints that we allow for is $C = \{\{0\}, \ldots, \{n-1\}\}$, represented by a state vector of zeros. We take it as the initial state of the MDP unless stated otherwise.

Actions – An action a consists of either adding or removing a constraint, driving the agent from one state to another. In practice, actions flip bits in the state vector corresponding to the encoded constraints. The agent is free to add a constraint of

any size, as long as the cost associated to the resulting set is within the computational budget. For instance, the agent can start by adding a 4-body constraint, e.g. ρ_{0123}, to the initial state. In contrast, removing a constraint has a different effect. In order to keep the state space exploration consistent, removing a constraint splits it into its most immediate components of a lower degree. For instance, in 1D, removing ρ_{0123} would result in ρ_{012} and ρ_{123}. Note that a valid action always corresponds to an arrow (in both directions) in the partially ordered set depicted in Fig. 4.2, remaining within the boundary (not crossing the orange dashed line).

Transition function – The transition function is a simple deterministic function implicitly defined above: $T(C|a, C')$ is a Kronecker delta, attaining unit value if the constraint configuration C is reached by adding or removing the constraint specified by the action a from the set of constraints C'.

Reward – We define the reward function to match the overall optimization goal, provided that the learning agent aims to maximize the obtained reward. The reward associated to a state C depends on: 1) the energy bound β_C, obtained solving its associated SdP (Eq. (4.8)), and 2) its computational cost. In practice, we take the amount of free parameters in the SdP, which we denote by p, as a representation of the computational cost. Note that, in general, given an optimization problem, we have no prior knowledge about the optimal β^* and its cost p^*. Therefore, in order to compute the reward, we rely on a set of references that are updated as the agent explores. More precisely, we keep track of the best and worst bounds obtained, β_{max} and β_{min} respectively, and the best and worst set of parameters with which the best bound β_{max} has been observed, denoted p_{best} and p_{worst} respectively. We compute the reward associated to a state by comparing its corresponding β and p to the reference values:

$$R(\beta, p) = \frac{p_{\text{best}}}{p_{\text{worst}}} \cdot \begin{cases} \frac{p_{\text{worst}}}{p} & \text{if } \beta = \beta_{\text{max}} \\ \left(\frac{\beta - \beta_{\text{min}}}{\beta_{\text{max}} - \beta_{\text{min}}}\right)^d & \text{otherwise,} \end{cases} \tag{4.9}$$

where d is a fixed exponent that controls the shape of the line $(\beta - \beta_{\text{min}})/(\beta_{\text{max}} - \beta_{\text{min}})$. We introduce this exponent to provide better discrimination among the higher bounds, typically $d = 5$. Notice that $p_{\text{worst}} \geq p_{\text{best}}$ and, therefore, $p_{\text{worst}}/p \geq 1$. Thus, the prefactor $p_{\text{best}}/p_{\text{worst}} \leq 1$ ensures that $R(\beta, p) \in [0, 1]$, $\forall\ \beta, p$. Fig. 4.3(b) shows a schematic of the reward function. In summary, the reward function mainly focuses on the resulting bound β, unless various states provide the maximum possible bound β_{max}. In this case, those with higher computational costs are penalized.

The agent – Within the proposed framework, we can perform the constraint optimization with various methods. As mentioned before, we propose to use RL with function approximation. The learning program or agent specifies the policy by which actions are taken with the ultimate goal of maximizing the obtained reward. More precisely, we use double deep Q-learning, introduced in Section 2.3.2. At each state C, the agent estimates the Q-values $Q^\pi(C, a)$ of each possible action a, a measure of the expected rewards associated with taking each action and then following the policy π. Then, it selects one according to an ϵ-greedy policy such that

$$\pi(C) = \begin{cases} \arg\max_a Q^\pi(C, a), & \text{with probability } (1 - \epsilon) \\ \text{uniform random } a, & \text{with probability } \epsilon. \end{cases} \tag{4.10}$$

Fig. 4.3 shows a schematic representation of the whole process. In Section 4.5 we show that such an approach leads to the optimal relaxation faster compared to other

classical optimization methods and, sometimes, it is able to find it even where the other methods fail.

4.5 Ground state energy bounds for the Heisenberg XX model

We apply the method described in the previous sections to find lower bounds to the ground state energy of quantum local Hamiltonians. To illustrate the process, we focus on a paradigmatic condensed matter model: the anti-ferromagnetic 1D quantum Heisenberg XX model [288], described by the Hamiltonian

$$H = \sum_{i=0}^{n-1} J_i(\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y) + \sum_{i=0}^{n-1} B_i \sigma_i^z \,, \tag{4.11}$$

where σ^α, $\alpha = x, y, z$, are the Pauli matrices, J_i is the antiferromagnetic exchange interaction between spins and B_i is the strength of the external magnetic field. We consider periodic boundary conditions, such that $\sigma_n^\alpha = \sigma_0^\alpha$. In the homogeneous case, $J_i = J, B_i = B \; \forall i$, the model presents a quantum phase transition at $B = 2J$ [103] between an antiferromagnetic and a paramagnetic phase, in which the entanglement vanishes [289–291]. We will hence refer to these phases as entangled and unentangled, respectively. Although the 1D XX model Eq. (4.11) is efficiently solvable via the Jordan-Wigner transformation [292], corresponding to a quadratic fermionic Hamiltonian [293–295], the agent is oblivious to such information. We emphasize that the points in the search space have no semantics to the agent, which, moreover, is not provided with any information about the Hamiltonian in any explicit way. This guarantees that our approach is as generally applicable as possible.

4.5.1 Optimal relaxations

Here, we present the results applying the RL method to the homogeneous version of the aforementioned Hamiltonian. For this case, we consider a computational budget that allows for the allocation of up to half of all the possible 3-body constraints. With these conditions, we find the best approximation to the ground state across the whole phase diagram of the Hamiltonian.

Unentangled phase ($B/J \geq 2$) – In the unentangled phase, the ground state can be perfectly described by the set of independent 1-body RDMs. Therefore, we would expect the optimal set of constraints to be the minimum that the agent can consider $C = \{\{0\}, \ldots, \{n-1\}\}$. Nevertheless, this is only true in the extreme case of $J = 0$. In a general scenario, with $0 < 2J \leq B$, the optimal set of constraints is made out of 2-body RDMs, as shown in Fig. 4.4 diagram (d). This is to provide support for the 2-body terms of the local Hamiltonian. Recall that, in our implementation, whenever a term H_i of the Hamiltonian is not supported by the set of RDMs $\Xi_C = \{\rho_S\}_{S \in C}$, we take $\langle H_i \rangle$ to be its minimal eigenvalue $\min(\sigma(H_i)) = -J$. With 2-body constraints, the resulting RDMs are rank-1 projectors, which correspond to pure states such that $\langle H_i \rangle = 0$ for the 2-body terms, thus yielding a better energy bound. Increasing the size of the constraints any further does not improve the energy bound at all.

Entangled phase ($B/J < 2$) – In the case of the entangled ground state, its exact energy can only be obtained by considering the system as a whole, corresponding to $C = \{[n]\}$. Therefore, the agent can only provide the best possible approximation to the exact energy within the allowed computational budget. Just like in the previous

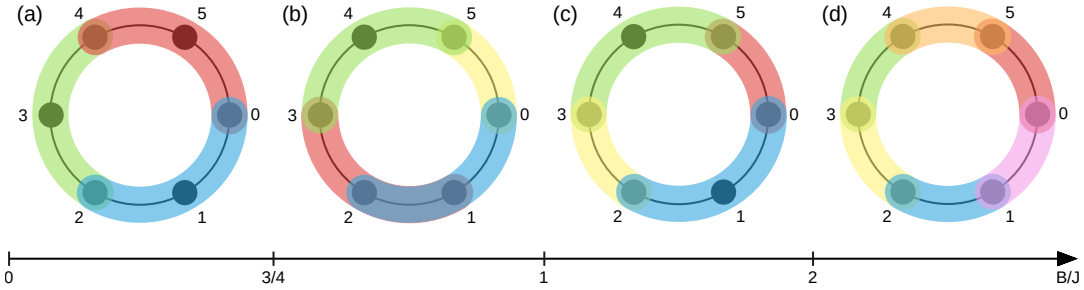

FIGURE 4.4: **Illustrative representation of the optimal constraints.** The optimal constraints C^* yield the best ground state energy bound with minimal cost for the homogeneous Heisenberg XX model in 1D (Eq. (4.11)). Each color represents the support of an RDM ($S \in C^*$) and we impose compatibility constraints over the overlapping areas. The results are obtained with a budget that allows for the allocation of up to half of the 3-body RDMs and $n = 6$ spins (black circles). For this case, the RL algorithm finds four different optimal solutions across the phase diagram. The expected relaxation is found deep into the entangled phase ($B/J < 2$) (a) at $B/J < 3/4$, displaying two intermediate solutions: (c) and (b), before the phase transition. The unentangled phase ($B/J \geq 2$) is solvable with a trivial relaxation (d) (see main text).

case, it may seem reasonable to expect the optimal set of constraints to be unique for the whole phase. Nevertheless, the agent finds three separate regimes as depicted in Fig. 4.4:

- Deep into the phase, as shown in Fig. 4.4 diagram (a), the best possible certificate is obtained by evenly distributing all the largest possible constraints throughout the system. A priori, we would expect this to be the optimal solution throughout the whole phase.

- In an intermediate regime, as shown in Fig. 4.4 diagram (b), the best possible certificate is obtained combining the overlap of some of the largest possible constraints with the inclusion of smaller constraints. It has the same computational cost as (a) but it provides a higher energy bound.

- Close to the phase transition, the optimal relaxation is obtained by alternating 2-body and 3-body constraints, as shown in Fig. 4.4 diagram (c). The resulting certificate has a lower computational cost than (a) and (b), but its energy bound is higher than (a) and the same as (b).

In the entangled phase, the two intermediate optimal relaxations (b) and (c) provide better bounds than the set of constraints (a) in Fig. 4.4, even with (c) being a significantly stronger relaxation than the others. In fact, (c) yields the exact same energy bound as (b) in the $1 \leq B/J < 2$ regime and its therefore optimal due to its significantly lower computational cost (see the appendix in Ref. [223] for the most technical details). Similarly, in the entangled phase, they all yield the same bound and thus the optimal is the simplest one, (d). This simple scenario shows that evaluating the quality of a relaxation beforehand is not a trivial task and it becomes even less straightforward when considering arbitrary Hamiltonians and computational budgets. Additionally, the given budget may also allow the allocation of a few 4-body RDMs, which we take into account in the optimization. However, the agent finds that it is better to combine several 3-body and 2-body RDMs rather than using a limited amount of 4-body ones.

Already in such a simple scenario, the agent is able to find a rich set of intermediate solutions, which may, at first glance, seem counter-intuitive. The solutions are,

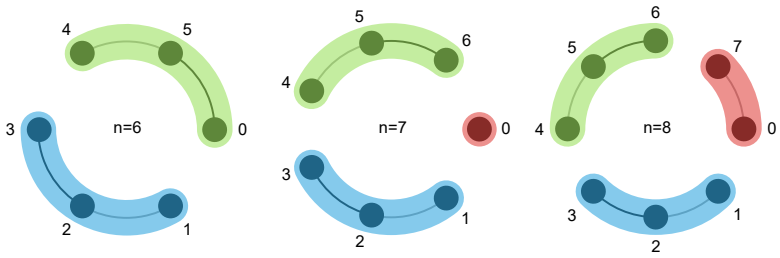

FIGURE 4.5: **Schematic representation of the inhomogeneous XX model with its optimal relaxations.** The figure shows examples for the three main paradigmatic setups using $n = 6, 7, 8$ spins. The external magnetic field is fixed, $B_i = 1\ \forall i$, and the interaction strength between spins, $J_i = i \mod 3$, is represented by the intensity of the black line: solid ($J_i = 2$), transparent ($J_i = 1$) and no line ($J_i = 0$). The support of the RDMs that constitute the optimal relaxation are depicted in different colours, as in Fig. 4.4.

nevertheless, closely related to the actual entanglement structure of the ground state of the system [290]. This shows that the agent is able to capture physical properties of the system, even when various possible solutions are very close in terms of cost and quality. In Fig. 4.4 we show the solution of a small system of $n = 6$ sites for illustrative purposes. In larger systems, we observe that the same optimal patterns remain consistent, suggesting that the qualitative solutions obtained in small systems can be used for larger ones with similar properties.

As a final remark, notice that the ground state of the unentangled phase is a product state, meaning that the exact solution lies within the budget with which the agent is provided. In contrast, in the entangled one, the ground state can only be exactly described by its full density matrix, meaning that the exact solution falls outside of the computational budget. With the framework we here present, when the agent is far from using the whole budget, it may be seen as a strong indication that the provided result is exact.

4.5.2 Comparison with other optimization methods

As we briefly mention at the end of Section 4.4, the proposed framework allows for the straightforward application of several optimization algorithms besides RL. In this section, we evaluate the quality of the RL results in comparison to two informative points of reference: breadth first search (BFS) [296] and Monte Carlo (MC) optimization [297].

For the comparison, we consider an inhomogeneous version of the XX Heisenberg model from Eq. (4.11) in which we keep a constant magnetic field $B_i = 1\ \forall i$ and tune the interaction strength $J_i = i \mod 3$. This provides us with isolated groups of three interacting sites. Note that, depending on the system size, there may be exclusively triplets, triplets and an isolated site or triplets and a pair, as we show in Fig. 4.5 for the cases of $n = 6, 7$, and 8, respectively.

Such model allows us to determine the optimal set of constraints beforehand while posing one of the hardest optimization instances, as it is a unique point in the constraint space. This way, we can compare the performance of the different algorithms with respect to the optimal solution. As performance metric, we compute the rewards, as in Eq. (4.9), with full knowledge of $\beta_{\max}, \beta_{\min}, p_{\text{best}}, p_{\text{worst}}$. This provides a measure of distance/similarity to the optimal configuration, obtaining reward 1 when reaching it.

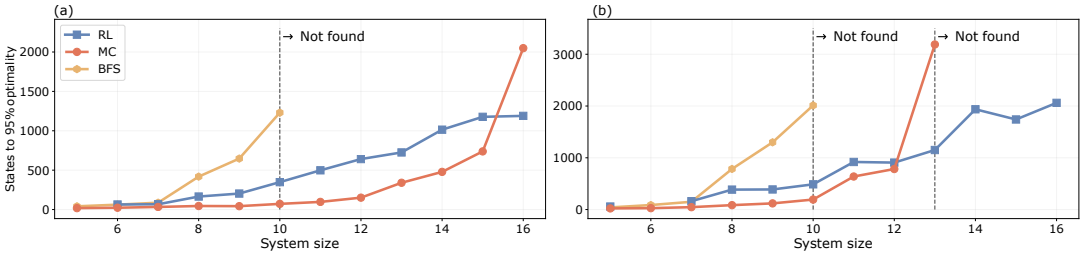

FIGURE 4.6: **Optimization benchmark in the inhomogeneous Heisenberg XX model.** Benchmark of the performance of the three optimization algorithms: BFS, MC and deep RL. The algorithms are evaluated in two scenarios: allowing up to (a) half of all the 3-body constraints and (b) all the 3-body constraints. The dashed vertical lines indicate the system size beyond which the overlapping algorithm was unable to find the optimal state in less than 4000 visited states.

Note that the algorithms have different ways to explore the state-space. Hence, in order to compare them in the fairest way, we do not take into account repeated visits to the states. Contrary to the BFS, both the RL and the MC agents can go back and forth revisiting the same states. Given that the main computational cost comes from solving the SdPs, we keep a memory of the solutions obtained during the exploration, so that revisiting a state implies a negligible computational cost.

Consequently, we evaluate the overall performance tracking the best obtained reward for every new visited state. In Fig. 4.6, we depict the amount of new states visited by fifty agents before they reach a reward of 0.95 on average. We repeat the process for several system sizes, with which the constraint space increases exponentially. We tune the hyper-parameters for the RL and MC optimizations at a system size of $n = 10$ and we keep them throughout the whole process.

First, we compare the agent performance with a budget that allows the agents to allocate half of all the available 3-body constraints. We show the results in Fig. 4.6(a). For small systems, there are no substantial differences in performance, given that the state space is reduced. Already at $n = 11$, the BFS is not able to find the optimal bound within a reasonable time. While the MC optimization provides better results for small systems, it is out-performed by the RL agent at $n = 16$. We hypothesize that, at this size, the overhead of learning is overcome by the increasing complexity of the state space.

In order to test this hypothesis, we conduct the same experiment with a larger computational budget that allows the agents to allocate all the 3-body constraints. With this, for the same system sizes, the agents encounter significantly larger constraint spaces. We show the results in Fig. 4.6(b). In this case, the differences between the MC and RL optimizations are relatively smaller for smaller systems and the RL agents outperform the MC optimization earlier on. This means that, for large state spaces, the learning cost involved in the RL optimization pays off, making it better than following a simple MC heuristic. In addition, unlike the RL, the MC shows a strong dependency on a proper hyper-parametrization, e.g., choosing an appropriate inverse temperature, provided that the performance is dramatically affected as soon as the parameters are not optimised for the specific problem. Proper parameter tuning is, in itself, a computationally costly task, given the constraint-space size. The RL scheme, being quite resilient to its hyper-parametrization, provides a significant advantage in this sense, allowing us to tune it in reduced systems.

4.5.3 Analysis with transfer learning across the phase space

An interesting feature of the proposed framework is that none of its parts require prior information about the actual problem. This suggests the possibility of exploring a given constraint optimization problem and its underlying system in a completely autonomous way. One way to take advantage of this feature is by performing transfer learning (TL) [298]. In order to do so, we start by training an agent to solve a system under the action of a Hamiltonian. Then, we leverage the experience obtained by the agent in the initial task using it as the initial condition to solve a new problem with a similar Hamiltonian.

For this task, we consider an homogeneous version of the Heisenberg XX model (Eq. (4.11)). As we have seen, while this Hamiltonian presents a unique quantum phase transition at $B/J = 2$, it has four different optimal relaxations across the phase diagram (see Section 4.5.1). We train an agent to find the optimal constraints deep in one phase, with $B/J = 5$. Then, we use the resulting trained agent to find the optimal relaxations for the rest of the phase space. In Fig. 4.7(a) we show the ratio between the time it takes the algorithm to converge with TL t_{TL} and the time it takes with a cold start t_0, i.e., starting from scratch. Hence, with $t_{TL}/t_0 < 1$ there is favorable TL and with $t_{TL}/t_0 > 1$ there is negative transfer. We obtain the convergence time averaging the training results of fifty independent agents, shown in Fig. 4.7(b).

We observe that TL in the same phase is significantly favorable. Indeed, for this particular problem, the optimal set of constraints is the same across the whole phase, including the critical point (cases (i) and (ii) in Fig. 4.7, respectively). When applied across phases, the advantage of TL diminishes sharply. Close to the phase transition (case (iii)), there appears a local minimum in which some agents get stuck and, under the given conditions, it takes them hundreds of training episodes to correct it. In this regime, the TL still provides an advantage regarding convergence, although it does not help avoiding the sub-optimal relaxation. Deep into the opposite phase (case (iv)), even though TL barely affects the performance, as $t_{TL}/t_0 \simeq 1$, it has a slightly negative impact.

The vertical lines of Fig. 4.7(a) show the phase transition (solid) and the intermediate points in which the optimal set of constraints changes (dashed), according to Fig. 4.4. The results suggest that losing the convergence advantage from TL can be indicative of changes in the ground state of the system, such as phase transitions. Hence, this approach can be used to infer properties of the physical system in a completely autonomous way by exploiting the failure of the method, such as in Refs. [105, 108].

4.6 Entanglement witnesses from the Heisenberg XX model

Here, we show how to apply the presented framework in the context of entanglement detection [299]. In particular, we address the task of building energy-based entanglement witnesses from local Hamiltonians [300, 301], a paradigmatic problem in quantum information processing. We illustrate how our constraint optimization framework benefits from being agnostic to the problem it is solving, as we only need to adapt the black box module that handles the SdP optimization from Eq. (4.8). Once we have that, we can readily apply the entire pipeline to the new problem.

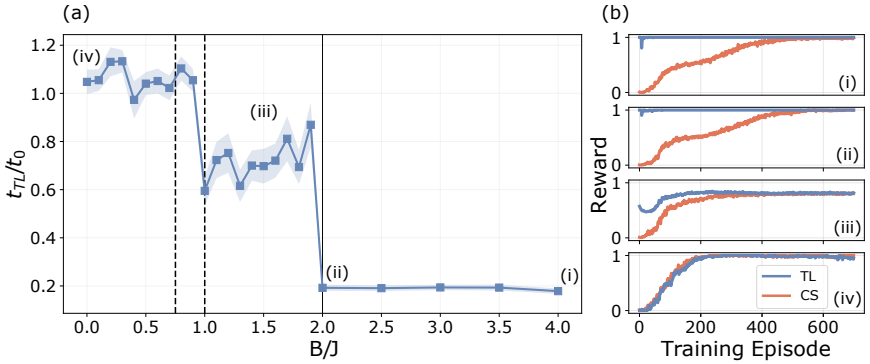

FIGURE 4.7: **Transfer learning results for an ensemble of fifty independent agents.** (a) Convergence time ratio between transfer learning and cold start as function of the parameter B/J (recall Eq. (4.11)). The pre-trained models are taken from $B/J = 5$ and they are used as starting point in the optimization for different values of the parameter. The vertical lines indicate qualitative changes in the optimal solution, with the solid line corresponding to the phase transition. (b) Average reward obtained at the final state of an evaluation episode, which is performed after each training episode, with TL and a cold start (CS).

4.6.1 Energy-based entanglement witnesses from local Hamiltonians

Entanglement is a fundamental property of quantum mechanics and it plays a key role in quantum information processing applications. However, characterizing or detecting entanglement in experimental applications can be hard due to the limited available information about the quantum state. A way to detect entanglement is through entanglement witnesses, which are observables whose expectation value can certify whether the state measured is entangled.

We can construct entanglement witnesses by choosing the observable to be the Hamiltonian itself. Let $\Delta E = \langle H \rangle - E_{\text{sep}}$ be our witness, where $\langle H \rangle$ is the expected energy of our state and E_{sep} is the minimum energy of the Hamiltonian in the set of separable states. This way, if $\Delta E < 0$, the quantum state lies outside of the separable set and, thus, it is entangled. Note, however that if $\Delta E \geq 0$, the witness does not decide.

To find E_{sep}, we need to solve a global optimization task of $\langle H \rangle$ over the set of separable states. The search can be restricted to pure product states via a convex roof argument, although that does not simplify the complexity of the optimization. herefore, we need to enforce that the global quantum state ρ is fully separable in Eq. (4.8). Nonetheless, even though the RDMs of a fully separable ρ are also separable, deciding membership to the set of separable quantum states is NP-hard [239], even in simpler instances [302–304].

We can relax this problem by considering the set of states that are PPT, which contains the set of separable states. These states are easy to characterize, as membership in the PPT set can be checked efficiently. However, they include some entangled states, thus being a relaxation of the set of separable states. Hence, we can modify Eq. (4.8), including the PPT constraints, to yield a lower bound on the separability bound of a local Hamiltonian H:

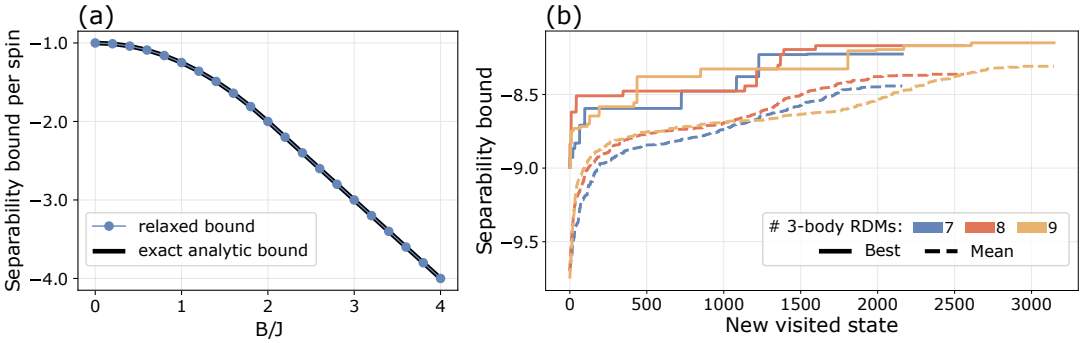

FIGURE 4.8: **Separability bounds obtained for the Heisenberg XX model.** (a) Resulting bounds for the homogeneous Hamiltonian in 1D compared to the exact ones across the phase space. (b) Separability bounds obtained by fifty independent agents with a random Hamiltonian and three different budgets: allowing 7, 8 and 9 3-body RDMs. The solid lines show the best bound among all agents and the dashed lines show their mean.

$$\begin{aligned} \beta_C^{\text{full}-\text{sep}} := \quad & \min_{\Xi_C} && \textstyle\sum_i \langle H_i \rangle && \\ & \text{s.t.} && \rho_S \succeq 0 && \forall S \in C \\ & && \rho_S^{\Gamma_A} \succeq 0 && \forall A \subseteq S \\ & && \text{Tr}[\rho_S] = 1 && \\ & && \text{Tr}_{R^c}[\rho_S] = \text{Tr}_{R^c}[\rho_{S'}] \quad \forall R \subseteq S \cap S', \quad && S, S' \in C, \end{aligned} \tag{4.12}$$

where the superscript Γ_A indicates that the partial transposition $(\mathbb{1}_{A^c} \otimes T_A)$ is applied to the elements of $S = A \cup A^c$. Note that this is a linear operation since it simply permutes elements of ρ_S. Thus, any quantum state satisfying $\text{Tr}[\rho H] < \beta_C^{\text{full}-\text{sep}}$ must contain some entanglement.

We can further tighten the optimization in Eq. (4.12) in several directions. For instance, we can consider symmetric extensions [235] to improve the approximation of the PPT set to the separable set. In some cases, we can ask that the bound detects a higher degree of entanglement, yielding a k-producibility bound. In this direction, there have been proposed device-independent witnesses of entanglement depth [305, 306], based on relaxations of the quantum marginal problem via an SdP. In these cases, we can tighten the relaxation by imposing compatibility with larger quantum states.

4.6.2 Optimal separability bounds

We start by considering an homogeneous version of the Heisenberg XX model in 1D, as in Section 4.5.1. In this case, there is an analytical expression for the separability bound, which corresponds to the mean-field ground state energy for negative J in cubic lattices [307]. We use this case as reference to validate our results. With the RL method, we find that we can recover the exact separability bound using exclusively 2-body RDMs, as we show in Fig. 4.8(a). Notice the separability bounds obtained in the unentangled phase ($B/J \geq 2$) coincide with the ground state energies obtained in Section 4.2.1, proving that the ground state is, indeed, separable.

Then, we consider an heterogeneous version of the Hamiltonian in a graph with random parameters, $J_i, B_i \sim \text{uniform}[0, 1)$. In Fig. 4.9(a), we show a schematic representation of the system. Unlike in the homogeneous case, here, there are no apparent

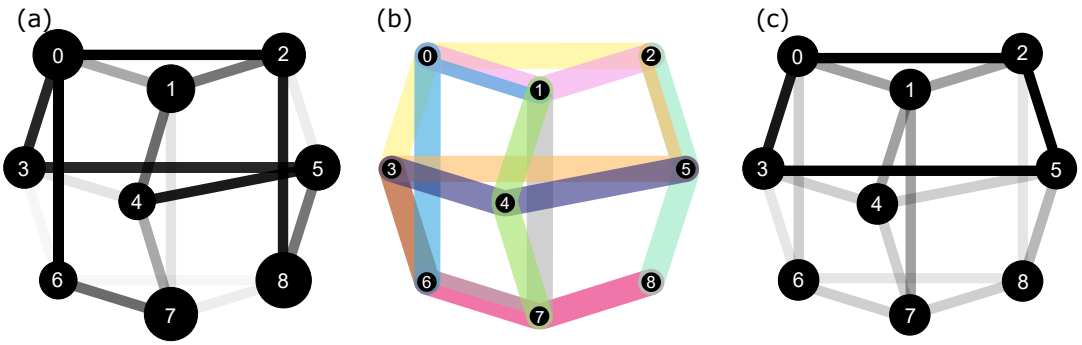

FIGURE 4.9: **Pictorial representation of the results obtained with a random Hamiltonian.** (a) Representation of the random Hamiltonian parameters. The line transparency and the circle radius respectively indicate the relative values of J_i and B_i. (b) Representation of the best constraints found by the agents. Each color represent the support of an RDM considered in the relaxation, as in Fig. 4.4. (c) Histogram of the terms contained in the best constraints of the top ten performing agents. The line transparency represents the term frequency normalized by the maximum.

properties of the system that we can exploit to obtain an expression for the separability bound. Furthermore, we cannot even devise a strategy to build an efficient relaxation, as we do in Section 4.5.2.

We search for the optimal relaxation with fifty independent RL agents with three different budgets: allowing seven, eight and nine 3-body constraints. We show the results in Fig. 4.8(b), where we see that we reach a higher bounds with every additional 3-body RDM. However, as the constraint space grows, it takes longer to reach the best configuration.

We depict the best obtained relaxation in Fig. 4.9(b), where we can see that the agent has significantly favoured some regions in detriment of others. For example, using a 2-body RDM in sites 3 and 7 despite having enough budget to span the whole system with 3-body RDMs. To see this more clearly, in Fig. 4.9(c), we represent the frequency with which each connection in the graph appears in the best constraints found by the top ten performing agents. This result highlights a few clear key elements to obtain better bounds, even though some of them may not seem relevant a priori given the Hamiltonian, such as the $\{2,5\}$ term, which has a weak interaction with strong fields on both sites. Looking further into these agents we found that the 3-body terms $\{0,1,2\}$, $\{2,5,8\}$, and $\{3,4,5\}$ are significantly overrepresented with respect to the rest, as they span the regions with the strongest interactions in the system.

4.7 Conclusion

In this chapter, have discussed a novel approach to construct optimal relaxations to obtain certificates of quantum many-body properties. We have proposed an ML approach, based on deep RL, to systematically find such certificates given a finite computational budget. Here, we have illustrated its properties in the context of approximating the ground state energy and the separability bounds of quantum local Hamiltonians.

We have studied the validity of the method in the well-known Heisenberg XX model, showing that the agent is able to correctly characterize the ground state across the phase diagram without any kind of information about the physical system at hand. Indeed, we have shown how the certificates found by the agent change

according to the ground state, as the structure of the constraints that suffice for a good approximation correlates with the system's phase and the entanglement properties of the ground state. Nonetheless, unravelling their precise relation is a matter deserving future investigation.

Already in small systems, we observe that the agent can capture the complexity of the system of study and go beyond more trivial relaxations, even when these are close in terms of the objective function. We have also shown that the agent is able to solve the opposite case, in which simpler certificates provide better bounds than more complex ones. Besides, in Ref. [223], we show that the qualitative relaxations obtained in reduced systems can be generalized to larger ones, as these remain consistent for any size. Hence, the constraint optimization can be performed in a reduced version of the original problem in order to minimize the computational workload.

Additionally, the RL approach handles large optimization spaces rather successfully, outperforming other classical optimization algorithms. Furthermore, we have shown how to take advantage of transfer learning, positively impacting scalability. Moreover, we have characterized its behaviour to find that it may be indicative of variations in the nature of the ground state of the system of study, some of which due to phase transitions. Hence, constituting an autonomous method to explore the system's phase diagram.

Finally, we have applied our framework in the context of entanglement detection. We have shown that we can efficiently recover the analytical solutions for simple cases and, as a final result, we apply our method to the case of a random Hamiltonian, to which there is no easy solution. The presented framework can be readily extended to other tasks in quantum information that are based on finding good outer approximations of convex sets that are hard to describe.

As future work, it remains open the question of which properties of the Hamiltonian have led to better bounds with simpler certificates. Furthermore, transfer learning can be used to analyze common patterns between different Hamiltonians. Besides, the architecture of the reinforcement learning agent can be adapted to allow for the transfer learning between problems of different sizes. As an additional step, it would be interesting to study how introducing explicit information about the Hamiltonian may affect the optimization process. For instance, whether a RL agent can help in designing better adiabatic schedules [234] or whether better certificates can be built by combining RL following an adiabatic path.

Chapter 5

Calibrating quantum computers with reinforcement learning

"This chapter is dedicated to Tom, Korbinian, and the Xanadu residents '23, who gifted me a wonderful summer understanding quantum computing and canoeing around Canada."

In this chapter, we introduce a reinforcement learning (RL) method to calibrate gates in quantum computers. This experimentally-friendly approach enables the consistent execution of high-fidelity quantum gates resulting directly from the interaction with the quantum computer. Therefore, it does not require any explicit modeling of the device, and is inherently scalable to large scale quantum computers and simulators, independently of their nature.

The content of this chapter is based on the implementation of a protocol inspired by Ref. [208]. Our contribution is detailed in a hands-on tutorial implementing the method in PennyLane [182] and JAX [308], accessible in Ref. [309].

5.1 Gates in superconducting quantum computers

Gate-based quantum circuits are the most common representation of quantum computations [162], as briefly introduced in Section 3.3. The circuits provide an abstraction layer that enables the research and development of quantum algorithms without delving into the specifics of the hardware in charge of the execution. However, each different quantum platform offers a unique set of interactions and controls that define the natural kinds of operations that can be executed in the hardware. These are commonly known as the *native gates* of the device, and they constitute the fundamental building blocks of any quantum algorithm executed in the device. Therefore, it is essential that such operations are executed as accurately as possible, which requires the careful tuning of the hardware's controls.

Here, we illustrate the process of finding the optimal control parameters using superconducting quantum computers as a case of study. Nevertheless, we note that the method presented in the following Section 5.2 is entirely hardware agnostic.

5.1.1 Single-qubit gates

In modern superconducting quantum computers, the qubits are coupled to wave guides that allow them to be selectively targeted by microwave pulses [310]. These pulses change the state of the qubits, effectively performing operations such as single-qubit rotations, which is usually referred to as *driving* the qubit.

Superconducting qubits are commonly modelled as a harmonic oscillators capacititatively coupled to a drive line [311]. Typically, only the first two levels of the

oscillator are considered and they are assigned to the computational basis states $|0\rangle$ and $|1\rangle$. These states have energies E_0 and E_1, respectively, resulting in a natural frequency of the qubit $\omega_q = (E_1 - E_0)/\hbar$. On the other hand, the driving microwave pulse can be written, without loss of generality, as a sinusoidal wave with time-dependent phase and amplitude $\Omega(t)\sin(\omega_d t + \phi(t))$. Here, $\Omega(t)$ and $\phi(t)$ denote the time-dependent amplitude and phase, respectively, and ω_d represents the frequency of the pulse.

Collectively, the dynamics of a driven qubit are captured by the time-evolution of an effective Hamiltonian with a constant or *drift* term H_0, and a time-dependent drive term H_d:

$$H(t) = \underbrace{-\frac{\omega_q}{2}\sigma^z}_{H_0} + \underbrace{\Omega(t)\sin(\omega_d t + \phi(t))\sigma^y}_{H_d}, \tag{5.1}$$

where $\sigma^{x,y,z}$ denote the Pauli operators. The drift term, H_0, indicates that the qubit is rotating around its axis at constant frequency ω_q. Therefore, it is helpful to change to a frame of reference that rotates along with the qubit in order to better understand the effect of the drive, H_d. The drive term in the rotating frame is

$$\hat{H}_d = \Omega(t)\sin(\omega_d t + \phi(t))(\cos(\omega_q t)\sigma^y - \sin(\omega_q t)\sigma^x)\,. \tag{5.2}$$

Rewriting $\sin(\omega_d t + \phi(t)) = \cos(\phi(t))\sin(\omega_d t) + \sin(\phi(t))\cos(\omega_d t)$, and letting $I = \cos(\phi(t))$ and $Q = \sin(\phi(t))$ denote the *in-phase* and *out-of-phase* components, respectively, we obtain:

$$\hat{H}_d = \Omega(t)(I\sin(\omega_d t) + Q\cos(\omega_d t))(\cos(\omega_q t)\sigma^y - \sin(\omega_q t)\sigma^x)\,. \tag{5.3}$$

Finally, proceeding with the multiplication, and removing the fast rotating terms that contain $\omega_q + \omega_d$ (known as the rotating wave approximation),

$$\hat{H}_d = \frac{1}{2}\Omega(t)\left((-I\cos(\delta\omega t) + Q\sin(\delta\omega t))\sigma^x + (I\sin(\delta\omega t) - Q\cos(\delta\omega t))\sigma^y\right), \tag{5.4}$$

where $\delta\omega = \omega_q - \omega_d$ is the frequency detuning between the qubit and the driving field. In the case of a resonant drive $\delta\omega = 0$, the expression is greatly simplified

$$\hat{H}_d = -\frac{1}{2}\Omega(t)\left(I\sigma^x + Q\sigma^y\right), \tag{5.5}$$

from where it becomes clear that controlling the pulse's phase $\phi(t)$, which determines I and Q, changes the axis around which the qubit is rotated by the pulse.

In Fig. 5.1(a), we simulate the time-evolution of a single-qubit superconducting quantum computer driven by different resonant pulses of the same duration. This allows us to illustrate the precise impact of the pulse's amplitude and phase on the resulting single-qubit rotation. On the one hand, increasing the amplitude with a fixed phase causes the the rotation axis to sweep vertically along the Bloch sphere, starting from the north pole at zero amplitude.[1] On the other hand, varying the phase at constant amplitude causes the rotation axis to spin around the Bloch sphere. Therefore, the interplay of both parameters enables the execution of any single-qubit rotation operation.

[1] In Fig. 5.1(a) and (b), we draw the positive eigenvectors of the resulting unitary operation, thus always remaining in the north hemisphere of the Bloch sphere. Nevertheless, the rotation axes traverse the entire sphere (negative and positive directions of the vector).

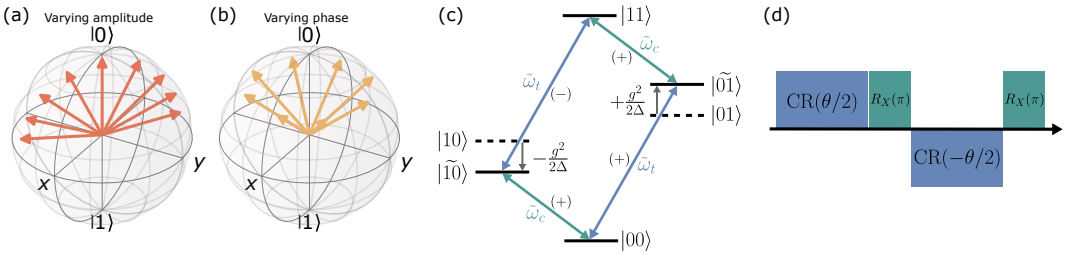

FIGURE 5.1: **Single- and two-qubit gates in superconducting quantum computers.** (a, b) Effect of the microwave pulse amplitude and phase, respectively, in the rotation axis of single qubit gates. (c) Energy spectrum diagram of two weakly interacting qubits. (d) Schematic representation of the train of pulses targeting the control qubit in an echoed CR gate. The colors indicate the frequency as in (c).

5.1.2 Two-qubit gates

Some qubits are connected to others via coupling buses, enabling the execution of entangling two-qubit gates between directly connected qubits. Among these, the cross-resonance (CR) gate [312, 313] is the most widely used in superconducting quantum computers provided that it operates exclusively with microwave pulses, and it is a precursor for the CNOT gate.

The CR gate capitalizes on a $\sigma^z\sigma^x$-type interaction that emerges between two weakly coupled qubits. For this reason, the CR gate is also known as the ZX or R_{ZX} gate. The gate is performed by shining a pulse on the first qubit (the *control*) at the second qubit's (the *target*) frequency. Depending on the state of the control, the target is rotated around the x-axis in the Bloch sphere in either the positive or negative direction. This behaviour becomes clear when inspecting its unitary matrix:

$$\mathrm{CR}(\theta) = \exp(-i\frac{\theta}{2}\sigma^z\sigma^x) = \begin{pmatrix} R_X(\theta) & \\ & -R_X(\theta) \end{pmatrix} \tag{5.6}$$

$$= \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) & 0 & 0 \\ -i\sin(\theta/2) & \cos(\theta/2) & 0 & 0 \\ 0 & 0 & \cos(\theta/2) & i\sin(\theta/2) \\ 0 & 0 & i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{5.7}$$

For $\theta = \pi/2$, the CR gate is a maximally entangling operation.

This phase acquired by the target qubit stems from the hybridization of the single-excitation energy levels, as shown in Fig. 5.1(c). Consider a control and a target qubit with frequencies ω_c and ω_t, respectively, resulting in a detuning $\Delta = \omega_c - \omega_t$. These qubits are weakly coupled at a rate $g \ll \Delta$, causing drives on the control qubit to be able to excite transitions in both the control and the target qubits. However, the drive amplitude on the target is damped by a factor g/Δ, resulting in intermediate hybrid states:

$$|\widetilde{10}\rangle \propto |10\rangle - \frac{g}{\Delta}|01\rangle \tag{5.8}$$

$$|\widetilde{01}\rangle \propto |01\rangle + \frac{g}{\Delta}|10\rangle\,. \tag{5.9}$$

The frequencies of the coupled qubits also shift accordingly with respect to their non-interacting frequencies resulting in $\tilde{\omega}_c$, $\tilde{\omega}_t$. This sign difference caused by the shift of the energy levels in opposite directions by $g^2/2\Delta$, is the origin of the negative phase when the control qubit is driven at the target's frequency $\tilde{\omega}_t$ [313], indicated by $(+)$

and $(-)$ in Fig. 5.1(c).

Nevertheless, there are multiple additional effects happening besides the $\sigma^z\sigma^x$ interaction between the qubits during the pulse [314]. For instance, while the control qubit receives an off-resonant pulse, it still induces a degree of single-qubit rotation in it, as it can be seen in Eq. (5.4). However, we can exploit the nature of the $\sigma^z\sigma^x$ interaction to isolate it. Instead of performing a single CR(θ) pulse, we split it into two pulses that rotate half the angle. Between the two CR pulses, we flip the state of the control qubit with a NOT gate, which is a rotation around the x-axis $R_X(\pi)$ performed with a resonant pulse. Then, we proceed with a *negative* second pulse, as shown in Fig. 5.1(c). Because the rotation direction in the target qubit depends on the state of the control qubit, flipping the control and applying the subsequent negative pulse is equivalent to performing a single full pulse. However, the second negative pulse "echoes out" the other unwanted interactions, such as the rotation in the control qubit [208, 314]. Hence, the name *echoed* CR. Finally, we apply a second NOT gate to the control to compensate for the previous one.

5.2 Gate calibration as a reinforcement learning problem

Superconducting quantum computers can perform single-qubit rotations and CR-type two-qubit operations, such as the CNOT, using microwave pulses. The execution of these gates relies on the careful selection of the pulse parameters: amplitude $\Omega(t)$, phase $\phi(t)$, frequency ω_d, and duration, which we collectively refer to as a *pulse program*. However, each qubit in the device has distinct properties, such as the frequency and the connectivity. These differences cause the same pulse programs to produce different operations for every qubit, even in the absence of noise. Consequently, every gate must be carefully calibrated for each individual qubit or qubit pair in the hardware.

A widely used strategy to derive the optimal pulse programs for quantum devices involves the detailed modelling of the system, enabling the gate optimization through analytical and numerical techniques [315]. Nevertheless, developing such a model requires an exhaustive characterization of the hardware. Even with comprehensive hardware knowledge, accounting for unknown transient Hamiltonian terms [316] or non-linear distortions of the drive fields and their couplings [317, 318] supposes a challenge. Furthermore, it is unfeasible to develop such highly accurate models for moderately large quantum computers, in practice, and simulations are restricted to relatively small systems, as discussed in Chapter 3.

An alternative approach to obtain the optimal pulse programs is through the direct interaction with the device, refraining entirely from deriving any explicit model of the system. Taking inspiration from the method proposed in Ref. [208], we frame qubit calibration as an RL problem. In this setting, an RL agent learns to calibrate the qubits in the quantum computer by tuning the control parameters and observing the qubits' responses. Through this process, the agent implicitly learns an effective model of the device, as it faces all the experimental nuances associated with the process, such as the effect of the most relevant noise sources.

A fundamental aspect of a robust calibrator its ability to react to the unique characteristics of each qubit and adjust the pulse parameters accordingly. To achieve this, the agent needs information about the intermediate states of the qubit(s) involved in the gate along the pulse. Since observing the qubit(s) states at any point during the pulse destroys the quantum state, we implement the following scheme:

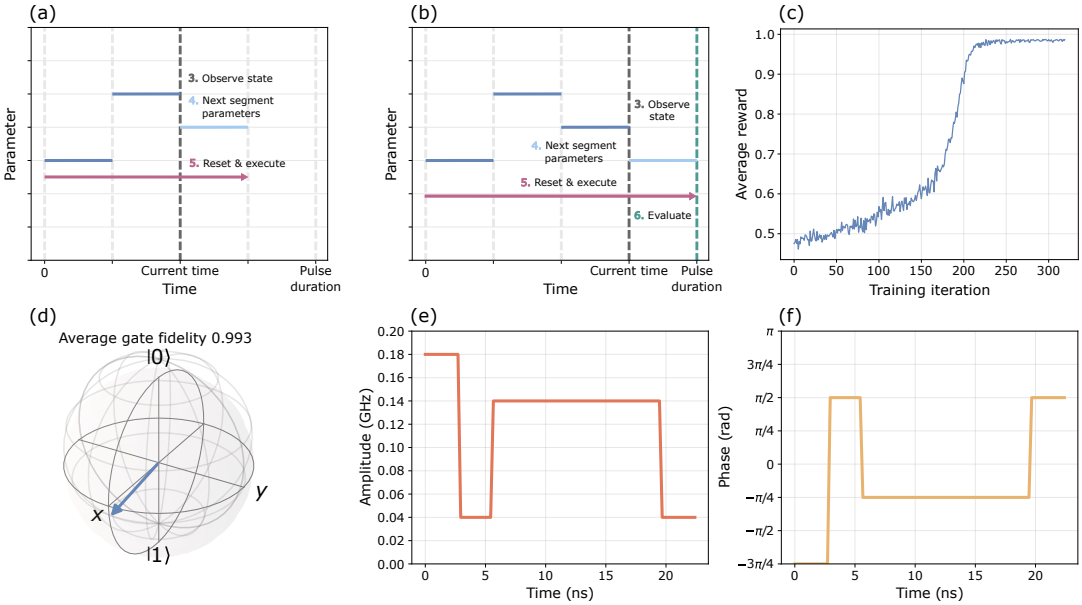

FIGURE 5.2: **RL calibration scheme and results.** (a, b) Schematic depiction of the main RL protocol steps. The numbers (from 3 to 6) correspond to the points in the protocol introduced in the main text. (c) Learning curve of an RL agent to calibrate an $R_X(\pi/2)$ gate. (d) Rotation axis of the resulting gate in the Bloch sphere. (e, f) Resulting piecewise constant (PWC) $\Omega(t)$ and $\phi(t)$ for the specific qubit.

1. Fix the pulse duration and split it into segments with constant parameters, commonly known as a piecewise constant (PWC) pulse.
2. Reset the qubit(s) involved in the gate.
3. Perform quantum tomography of the qubit(s) to determine the state.
4. Based on the information, the agent chooses the parameters of the next segment in the PWC pulse.
5. Reset the qubit and execute the pulse up to the last segment with fixed parameters.
6. Repeat steps 3 to 5 until the end of the pulse is reached and evaluate the average gate fidelity.

Fig. 5.2(a) and (b) show a schematic representation of the main points in the protocol. In Fig. 5.2(a), the first two pulse segments (dark blue) are executed as a shorter pulse than the final result. This is repeated several times to perform quantum state tomography of the state, whose result is used to determine the parameters of the next pulse segment (light blue). Afterwards, the system is reset and the first three segments are executed, which leads to Fig. 5.2(b). There, the steps are repeated in the same order, evolving the qubit with the first three segments several times to characterize their effect and determine the parameters of the fourth segment, reaching the end of the pulse. Given that all the pulse segments are now fixed, we proceed with the evaluation of the pulse with respect to the target gate that we wish to execute. Overall, the agent iteratively builds a PWC pulse that is tailored to the specific qubits. Even though this protocol involves multiple intermediate tomography steps, the overall cost is relatively low provided that it is only for the qubits involved in the gate, typically one or two.

In RL terms, the environment is the actual quantum computer the agent interacts with. The states can be any information about the intermediate qubit states after each segment of the PWC pulse, from a direct reconstruction of the state with full tomography, to a collection of relevant observables.

The actions adjust the pulse program parameters. Out of those, point 1 in the protocol fixes the maximum pulse length, and we always consider resonant pulses with the target qubit, thus fixing ω_d. This allows the agent to tune the amplitude $\Omega(t)$ and the phase $\phi(t)$ for every segment. For simplicity, we consider a discrete set of values for both and let every action denote a combination of phase and amplitude values. Hence, given an observation, the agent's action directly determines the pulse parameters for the next segment. However, it is worth noting that this approach has strong limitations in the amount of discrete values that can be considered for every parameter and, in some cases, it may be more convenient to consider continuous actions.

Finally, the reward is a function of the average gate fidelity. We sample several initial random states and apply the pulse program several repeated times. Then, we compute the average fidelity between the resulting intermediate and final states from our pulse, and the expected states from the target gate. Finally, we take the weighted average of all the fidelities, giving more relevance to the initial gate applications. This process of repeatedly applying the pulse programs accumulates the errors, making our reward function more sensitive to them. This allows the agent to better refine the pulse programs as other error sources, such as measurement errors, become less relevant with more repetitions. While we have mostly focused on the simulation of ideal quantum computers, the reward can incorporate additional terms to obtain better results in experimental scenarios, such as a penalty for driving the qubits beyond the first excited state (e.g. reaching $|2\rangle$).

We train the agent with the REINFORCE [82] algorithm using the optimal state-independent baseline, as introduced in Section 2.3. Every episode is a full pulse program execution and evaluation, building the PWC pulse segment by segment from the qubit initialization. In Fig. 5.2(c) to (f), we show the results of training a calibrator for an $R_X(\pi/2)$ gate (or $\sqrt{X}$) in a single-qubit device. Even though the average training reward plateaus around 0.986, the final average gate fidelity is 0.993, respectively shown in Fig. 5.2(c) and (d). The pulse program has eight segments and a total duration of 22.4 ns, significantly shorter than the standard 36 ns pulses [319]. Fig. 5.2(e) and (f) show the resulting PWC values for $\Omega(t)$ and $\phi(t)$, respectively, whose intricate interplay enable the realization of shorter pulses with high fidelity.

Part II

ML for diffusion: from proteins in cells to internet users

Chapter 6

Machine learning for diffusive processes

In this second part of the thesis, we focus on the applications of machine learning (ML) techniques to study stochastic processes. In this introductory chapter, we provide an introduction to the fundamental concepts of diffusion, as well as an overview of ML applications to the field. Then, we introduce our contributions to the topic in Chapters 7 to 9.

6.1 Normal and anomalous diffusion

Random walks describe a trajectory formed by random steps in a mathematical space [320]. They are employed to describe phenomena across diverse scales and scientific domains, such as the motion of atoms in periodic potentials [321] or molecules and proteins in cells [322–324], the navigation patterns of animals in their habitat [325], the dynamics of stock prices [326], or even the evolution of our personalities over time [327].

A quintessential example of a random walk is Brownian motion, which describes the movement of particles suspended in a medium. This phenomenon was originally observed by Robert Brown as he studied the motion of pollen particles in water early in the nineteenth century [328]. Nearly a century later, Einstein and Smoluchowski formalized the movement of the particle as a result of its collision with the smaller surrounding particles of the medium [329–331]. Attempting a deterministic description of this phenomenon would require to account for an overwhelming amount of degrees of freedom, on the order of the Avogradro's number $\sim 10^{23}$. Consequently, Brownian motion was formalized as a stochastic process where the probability of finding the particle in position x at a given time t follows the diffusion equation:

$$\frac{\partial}{\partial t}P(x,t) = D\frac{\partial}{\partial x^2}P(x,t)\,. \tag{6.1}$$

Here, D is the *diffusion coefficient*, a parameter that encompasses all the relevant physical properties of the system, including the temperature, the medium viscosity, and the mass, size, and nature of the diffusing particle. Considering that the particle starts at the origin, the solution to Eq. (6.1) yields a Gaussian probability density function with zero mean and standard deviation $\sigma = \sqrt{2Dt}$:

$$P(x,t) = \frac{1}{\sqrt{4\pi Dt}}\exp\left(-\frac{x^2}{4Dt}\right)\,. \tag{6.2}$$

This distribution implies a null first moment $\mathbb{E}\left[x(t)\right] = 0$, indicating the particle has an equal probability of moving in any direction. However, the second moment

grows linearly with time

$$\mathbb{E}\left[x^2(t)\right] \propto 2Dt\,. \tag{6.3}$$

This is known as the mean squared displacement (MSD), and it provides an intuition of the space the particle has explored over time. Its linear temporal dependence is characteristic of *normal diffusion*.

In an experimental setting, the position of a particle $x(t)$ is typically sampled at a finite frequency, resulting in a succession of random displacements Δx separated by constant time intervals. In these cases, we recover the probability density function from Eq. (6.2) by the central limit theorem, assuming the displacements are independent and identically distributed with finite variance, and the time intervals have finite mean [332, 333]. This formulation in terms of displacements and time intervals is very convenient to analyze the trajectories and develop theoretical models to describe them, as we illustrate in the upcoming Section 6.2. Indeed, the probability of performing a displacement Δx in a time interval Δt is analogous to the probability of observing the particle at position x at time t from Eq. (6.2).

However, in many physically-relevant situations, the central limit theorem may not apply due to long time correlations, fat tails in their distributions, heterogeneity in the system, or waiting times with diverging mean [334]. These factors lead to deviations from normal diffusion, known as *anomalous diffusion*, which may result in non-Gaussian probability density functions and/or a non-linear MSDs dependence with time:

$$\mathbb{E}\left[x^2(t)\right] \propto 2D_\alpha t^\alpha\,, \tag{6.4}$$

where α is the *anomalous diffusion exponent*, and D_α is an effective diffusion coefficient. We encounter two main regimes depending on α: subdiffusion with $0 < \alpha < 1$, and superdiffusion with $1 < \alpha < 2$. For $\alpha = 1$, we recover normal diffusion, and the extreme cases of $\alpha = 0$ and 2 correspond to immobile and ballistic trajectories, respectively.

Traditionally, the most common way to characterize normal and anomalous diffusion is by fitting the MSD over time. For normal diffusion trajectories, the slope is proportional to D. Analogously, in anomalous diffusion trajectories, α corresponds to the slope in logarithmic space $\log(MSD) \propto \alpha \log(t)$.

To estimate the MSD, we can consider an ensemble of N independent realizations of the random walk, and compute the average displacement made by the particle in a time window t_{lag}:

$$\text{EA-MSD}(t_{\text{lag}}) = \frac{1}{N}\sum_{i=1}^{N} x_i(t' + t_{\text{lag}}) - x_i(t')\,. \tag{6.5}$$

This is known as the ensemble-averaged MSD (EA-MSD) and, in practice, we often take $t' = 0$. However, we may not always have access to sufficient realizations of the same random walk. For example, it is common that individual trajectories in the same experiment display different behaviours. In these cases, we can consider the displacement averaged along the individual trajectories. For a trajectory composed of T frames, the so-called time-averaged MSD (TA-MSD) would be computed as:

$$\text{TA-MSD}(t_{\text{lag}}) = \frac{1}{T - t_{\text{lag}}}\sum_{t'=0}^{T-t_{\text{lag}}} x(t' + t_{\text{lag}}) - x(t')\,. \tag{6.6}$$

Here, we let T, t' and t_{lag} denote the number of time frames. To estimate D and α, we evaluate either the EA-MSD or TA-MSD for different time windows t_{lag}, and

proceed with a linear fitting of the result.

Finally, the combination of both EA-MSD and TA-MSD provides insights into the ergodicity of the underlying process. The ergodic theorem, introduced by Boltzmann [335], states that a process is ergodic if a single realization can explore all the possible configurations of the system. Ergodicity breaking is usually related to the existence of mutually inaccessible domains with effectively-infinite barriers that separate the phase space. In these cases, a single trajectory cannot faithfully describe the properties of the system, provided that it has only visited a limited set of states. However, when those mutually exclusive regions are all accessible, an ensemble of trajectories can explore the entire phase space resulting in weak ergodicity breaking [336]. This results in a discrepancy between the EA-MSD and the TA-MSD, which is no longer a valid measure of the properties of the system [334].

6.2 Theoretical models to describe anomalous diffusion

Given the predominant presence of anomalous diffusion across a wide variety of fields, there have been considerable efforts to unveil and describe its underlying mechanisms [334]. In this section, we briefly introduce five prominent anomalous diffusion models, each capturing different phenomenologies.

6.2.1 Continuous time random walk

The continuous time random walk (CTRW) [337] is a model particularly well-suited to describe the motion of particles susceptible of being trapped or immobilized [338].

In its discrete version, the diffusing particle waits a random amount of time Δt_i before performing each step Δx_i. When the time arrives, the steps are instantaneous and their size is independent of the *waiting time*. The waiting times are sampled according to a distribution $\psi(t)$, and our focus in this thesis is on cases where $\psi(t)$ exhibits a long tail, such as a power-law, introducing a non-zero probability of an infinite waiting time. This results in subdiffusive behaviour with weak ergodicity breaking, provided that a particle with an infinitely long waiting time will never explore the entire space.

The anomalous diffusion exponent α is directly determined by the waiting time distribution:

$$\psi(t) \propto t^{-\alpha-1} \tag{6.7}$$

for $0 \leq \alpha \leq 1$. Notably, this model converges to Brownian motion for $\alpha = 1$.

6.2.2 Lévy walk

Lévy walks (LWs) [339] are similar to CTRWs, although, in this case, the size of the displacements is correlated with the waiting times. They find multiple applications in physics, biology, and optimal search strategies [340], such as those followed by foraging animals [325].

In contrast to CTRWs, where the particles instantly jump once the waiting time is over, LWs take the path into account.[1] In their simplest form, the particles move in a straight line at constant velocity v for the entire waiting time Δt, covering a distance $\Delta x = v\Delta t$. At the end of the excursion, the particles randomly choose

[1] Lévy *flights* are analogous to CTRW in the sense that they also feature discrete jumps after each waiting time. However, unlike CTRWs their displacement size is correlated to the waiting time, just like Lévy walks.

another direction and move for another random period of time $\Delta t \sim \psi(t) \propto t^{-\sigma-1}$ at the same speed. This results in a coupled transition probability:

$$\Psi(\Delta x, \Delta t) = \frac{1}{2}\delta\left(|\Delta x| - v\Delta t\right)\psi(\Delta t)\,, \tag{6.8}$$

where the Dirac delta δ indicates the particle stops to change its velocity direction only when the flight time Δt is over and it has covered a $|\Delta x|$ distance.

With this formulation, $\alpha = 2$ for $0 < \sigma < 1$, and $\alpha = 3 - \sigma$ for $1 < \sigma < 2$, making the model inherently superdiffusive. Furthermore, as there can be infinite flight times, LWs show weak ergodicity breaking as well.

6.2.3 Annealed transit-time model

The annealed transit-time model (ATTM) [341] aims to mimic the spatio-temporal heterogeneities inherent in biological environments [324].

With this goal, ATTM describes the motion of a Brownian particle whose diffusion coefficient D undergoes random changes over time. The particle initiates its motion with a random D, which remains constant for a random *dwell time* τ. Then, it transitions to another random D' and continues to diffuse for another dwell time τ'. This process is repeated throughout the whole trajectory changing diffusion coefficient after each dwell time. The diffusion coefficients are sampled according to a probability distribution that follows a power-law behaviour $P(D) \sim D^{\sigma-1}$, $\sigma > 0$ for $D \to 0$, and a fast decay for $D \to \infty$. The dwell times for each D are sampled from a conditional probability distribution with expectation value $\mathbb{E}\left[P(\tau|D)\right] = D^{-\gamma}$.

Even though the particle undergoes normal diffusion at short time scales, the diffusion is anomalous and weakly non-ergodic at long time scales. The anomalous diffusion exponent α is directly determined by the the probability distributions $P(D)$ and $P(\tau|D)$. In the regime that we consider in this thesis, with $\sigma < \gamma < \sigma + 1$, ATTM is subdiffusive with $\alpha = \sigma/\gamma$.

6.2.4 Fractional Brownian motion

Fractional Brownian motion (FBM) [342, 343] can describe the motion of particles that exhibit long-time correlations, from confined particles, to actively-moving ones [344, 345].

It can be derived from the Langevin equation [346], which stems from Newton's second law, in the presence of fractional Gaussian noise[2] ξ_{fGn}:

$$\frac{dx(t)}{dt} = \xi_{fGn}\,(t). \tag{6.9}$$

The noise has zero mean and power-law self-correlation function:

$$\mathbb{E}\left[\xi_{fGn}(t_1)\xi_{fGn}(t_2)\right] = \alpha(\alpha-1)D_\alpha|t_1 - t_2|^{\alpha-2}\,, \tag{6.10}$$

for $t_1 \neq t_2$, which results in correlated displacements:

$$\mathbb{E}\left[\Delta x(t_1)\Delta x(t_2)\right] = D_\alpha\left(t_1^\alpha + t_2^\alpha - |t_1 - t_2|^\alpha\right)\,. \tag{6.11}$$

[2] Brownian motion can be derived in the same way using white noise, as an alternative to Eq. (6.1).

The factor $(\alpha - 1)$ indicates the noise is persistent for $1 < \alpha < 2$, and anti-persistent for $0 < \alpha < 1$. The former results in positively correlated displacements, characteristic of directed motion, that lead to superdiffusion. Meanwhile, the latter produces anti-correlated displacements, characteristic of bouncing particles, that lead to subdiffusion. In the limit of $\alpha = 1$, the noise becomes uncorrelated, resulting in standard Brownian motion.

Regardless of α, FBM is ergodic, unlike the rest of the anomalous diffusion models presented throughout this Section 6.2. As a final remark, FBM is commonly studied in terms of the Hurst exponent $H = \alpha/2$.

6.2.5 Scaled Brownian motion

Scaled Brownian motion (SBM) [347, 348] describes the motion of particles whose diffusivity changes smoothly with time. This can capture spatio-temporal heterogeneities of the environment, such as temperature changes over time, or the presence of concentration gradients and molecular crowding [349–351].

SBM can be derived from the Langevin equation, similar to how FBM is formulated from Eq. (6.9). This time, however, in the presence of white noise with a time-dependent variance:

$$\frac{dx(t)}{dt} = \sqrt{2D(t)}\xi(t)\,, \tag{6.12}$$

where $\xi(t)$ is uncorrelated Gaussian noise with zero mean and unit variance, such that $\mathbb{E}\left[\xi(t_1)\xi(t_2)\right] = \delta(t_1 - t_2)$. The diffusion coefficient follows a power-law dependence with time $D(t) \propto D_\alpha t^{\alpha-1}$.

For $\alpha < 0$, the diffusion coefficient decays steadily, resulting in an ever-slower motion that leads to subdiffusion and weak ergodicity breaking. Conversely, $\alpha > 1$ results in an accelerated motion that leads to superdiffusion. In the case where $\alpha = 1$, the diffusion coefficient remains constant and we recover the standard Brownian motion.

6.3 Machine learning to study anomalous diffusion

Characterizing diffusion processes is crucial to understand the complex underlying physical and biological mechanisms governing them. This characterization often involves a thorough analysis of experimental data obtained from single-particle tracking experiments. In these experiments, individual tracers are monitored for a period of time over which they describe trajectories in their medium. As mentioned earlier, the tracers can take multiple forms, from animals to more abstract elements such as stock values. Studying their trajectories, we can extract meaningful parameters, such as the diffusion coefficient D or the anomalous diffusion exponent α, that provide valuable insights into the inherent characteristics of the tracers and their medium. Additionally, we can assess which diffusion model, such as those introduced in Section 6.2, best describes the observations to gain further information about the phenomenology of the system.

However, accurately capturing and analyzing the trajectories presents several challenges at multiple levels. The combination of their stochastic nature with several experimental challenges, such as imaging noise or the tracer's photophysics, is added on top of the complex dynamics characteristic of anomalous diffusion [334, 352]. In Section 6.1, we have introduced the EA-MSD and TA-MSD to characterize diffusion processes. While fitting the MSD offers an optimal way to estimate

D for trajectories undergoing Brownian motion in many circumstances [353, 354], as we show in Chapter 8, studying anomalous diffusion trajectories is inherently more challenging and prone to artifacts associated to the presence of experimental noise [355]. For these cases, fitting the EA-MSD can provide a faithful estimation for α when we have access to an ensemble of trajectories [356]. In the case where the ensemble presents heterogeneous behaviours, the individual information of each trajectory can be obtained fitting the TA-MSD, although it is limited to ergodic and sufficiently long trajectories [357], as presented earlier. Nevertheless, estimating α from the MSD can introduce significant errors and biases. The accuracy of the estimation is very vulnerable to stochastic fluctuations, requiring large amounts of data to compensate, either in the form of more or longer trajectories. Acquiring long trajectories is especially important, as the asymptotic behaviour of the MSD may differ from that at short times, requiring large t_{lag} in Eqs. (6.5) and (6.6) to obtain a correct estimation of α [334]. However, access to such large amounts of data is typically restricted due to experimental constraints. Furthermore, the estimation of α is biased by noise, such as the localization precision of the experiment [355, 358]. Therefore, the experimental noise needs to be characterized in parallel in order to correct the final value [357].

Among the plethora of alternative methods, such as those based on the power spectral density [359, 360] or Bayesian estimation [361], ML-based approaches stand out as powerful and flexible tools to study diffusion processes in a wide variety of cases [362]. Already the first approaches, developed between 2019 and 2020, showed that these models not only provide accurate estimates for α, but they also enable the classification of the trajectories into the phenomenological models that best describe them [363–366]. The radical differences between approaches, stemming from the sheer amount of possibilities offered by the field of ML, motivated the organization of the *AnDi Challenge* [367]. As we detail in the upcoming Chapter 7, the AnDi challenge is a competition to characterize anomalous diffusion that provides a common ground to develop and compare different methods. Even though there was no absolute winner of the challenge, methods based on deep learning (DL) applied either directly to the raw data, or in combination with feature engineering, dominated the competition [368–372]. This competition has instigated the research for the development of new DL-based techniques, mostly employing convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which have already proven instrumental for the analysis of complex phenomena in different experimental settings [373, 374]. Further developments focus on the research of reliable and interpretable methods that provide a measure of uncertainty for the estimates [375], or a better understanding of the underlying physics [376]. Other research avenues strife to incorporate the latest advances in the field of ML to the analysis of diffusion processes [377, 378]. Finally, despite our focus on the analysis of trajectories, there is a noteworthy effort in the development of ML algorithms to improve the quality of the acquired data, from designing better experimental hardware [147], to enhancing particle-tracking algorithms [379].

Thus, ML techniques play an increasingly important role in the analysis of diffusion processes. In Chapter 7, we detail our contribution to the AnDi challenge, and we introduce a novel method to study diffusion processes in Chapter 8. Finally, in Chapter 9, we illustrate an industrial application analyzing the trajectories of users in the action-space of the web browsing.

Chapter 7

HYDRA: characterizing anomalous diffusion

In this chapter, we will dive into a specific machine learning (ML) method designed to characterize anomalous diffusion at a single-trajectory level. This method constitutes our contribution to the first edition of the AnDi challenge [367], where it was presented as part of the efforts of the *Anomalous Unicorns* team. The code with detailed explanations can be found in Ref. [380]. Furthermore, we have contributed to the development of the `andi_datasets` python library [381], which is used to generate the data sets in the AnDi challenge and it currently stands as a valuable resource for ongoing research on the field.

7.1 The AnDi Challenge

The AnDi challenge [367] is a competition dedicated to finding the most effective methods for the analysis of anomalous diffusion processes. The challenge provides an even testing field to develop and compare methods in physically relevant scenarios, fostering a communal effort that allows the research community to collectively advance the field. The inaugural edition of the AnDi challenge, held in 2020 [367], concentrated on the theoretical exploration of anomalous diffusion, putting the models introduced in Section 6.2 at the center. The forthcoming second edition, scheduled for 2024 [382], shifts the focus to phenomenological models, emphasizing the discovery and characterization of diffusion changes, aligned with the topics we present in Chapter 8.

Here, we focus on the first edition of the challenge. The challenge proposes three different tasks to characterize anomalous diffusion at the single-trajectory level:

Task 1 Infer the anomalous diffusion exponent α.

Task 2 Infer the anomalous diffusion model that best describes it.

Task 3 Find a unique change point in the trajectory and provide α and the diffusion model for each part.

All the trajectories in the challenge were simulated based on the anomalous diffusion models detailed in Section 6.2. Consequently, the tasks involving a classification of the models need to output one of these five models. Additionally, every task is further subdivided into three categories, corresponding to trajectories in one, two, and three dimensions, resulting in a total of nine tasks.

The challenge received contributions from multiple teams emplying a wide spectrum of different strategies. The most traditional approaches focused on extracting meaningful features from the data [383], and performing Bayesian inference of

the motion parameters [384, 385]. However, most teams leaned towards ML-based approaches. DL methods were predominantly applied directly to raw data [369–371], while other ML schemes, such as extreme learning machines or gradient boosting, relied on feature engineering [386, 387]. Notably, the combination of both approaches yielded remarkable results results [368, 372]. Even though there was not a clear overall winner of the competition, DL-based solutions provided the best results overall.

The outcome of the AnDi challenge has not only influenced the research of methods to study anomalous diffusion, but it also has established itself as a community-driven benchmark for the evaluation of novel algorithms, exemplified by Ref. [388] or our contribution discussed in Chapter 8. Furthermore, the python package to simulate anomalous diffusion trajectories developed for the challenge, `andi_datasets` [381], serves as a valuable a tool to aid the research on the field [375, 378, 388, 389].

7.2 HYDRA: a modular feature extractor

As we detail in Section 7.1, the central objective of the AnDi challenge is to characterize anomalous diffusion trajectories at a single-trajectory level. Within the context of the challenge, this mainly involves two aspects: determining the anomalous diffusion exponent α (Task 1), and identifying the diffusion model that best describes the trajectory (Task 2). Our contribution to the challenge focussed on addressing these two tasks for 1-dimensional trajectories. Interestingly, Task 3 of the challenge, which centers around finding changes in diffusion properties along trajectories, served as an inspiration for the work presented in Chapter 8.

From the perspective of ML, inferring α can be directly formulated as a regression task. Conversely, finding the anomalous diffusion model that best describes the trajectory can be framed as a classification task. Since all the trajectories in the competition are simulated with the `andi_datasets` library [381], this involves determining the diffusion model used to generate the trajectory. The potential models include: annealed transit-time model (ATTM) [341], continuous time random walk (CTRW) [337], fractional Brownian motion (FBM) [342, 343], Lévy walk (LW) [339], and scaled Brownian motion (SBM) [347], all of which are detailed in Section 6.2.

The diffusion models exhibit different characteristics, which can lead to significant differences in the generated trajectories between one another, with their main features manifesting in different forms and time-scales. Building upon the previous success of recurrent neural networks (RNNs) and convolutional neural networks (CNNs) in the characterization of anomalous diffusion [363–365], we propose a modular architecture with various feature extractor heads that converge into a single body, hence the name *HYDRA*. The main building block is a two-headed HYDRA, featuring an RNN and a CNN feature extractors that converge into a fully-connected block, illustrated in Fig. 7.1(a). This design allows the model to capture both long- and short-range correlations in the trajectories, which enables it to identify the specific characteristics inherent to the anomalous diffusion models. These bi-headed modules can be stacked in parallel to conform multi-headed HYDRAs, as we detail in the subsequent Section 7.3.

More precisely, the RNN blocks employ regularized gated recurrent unit (GRU) layers [64] that process the input trajectory sequentially, as explained in Section 2.2.2. The CNN components are built using XResNet blocks [61], as detailed in Section 2.2.2. This architecture enables the model to handle trajectories of variable lengths, a crucial feature for any real-world experimental scenario. In the challenge,

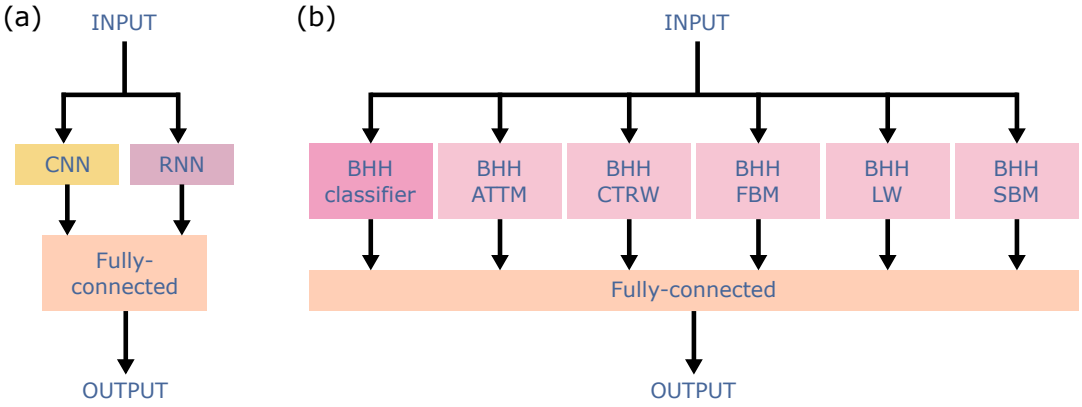

FIGURE 7.1: **Hydra architecture.** (a) Bi-headed HYDRA architecture. The input is processed in parallel by a CNN and an RNN, whose outputs are concatenated and passed through a fully-connected NN. (b) Mulit-headed hydra employed for the anomalous diffusion exponent regression. The architecture features 6 bi-headed HYDRAs (BHH in the figure): a classifier, and an expert for each of the 5 anomalous diffusion models. Their outputs are concatenated and processed by a fully-connected NN.

this is reflected by generating trajectories with between 10 to 1000 points in total. Given the potentially large length disparities between trajectories, we sample the trajectory batches pseudo-randomly during training to ensure that trajectories within the same batch have similar lengths, thereby substantially speeding up the training process by minimizing wasted computation. Finally, the resulting models for both the trajectory classification and regression are ensembles of ten HYDRAs. These ensembles consist of independent models trained for the same task that make joint decisions: for regression, we calculate the mean prediction of all the models, and for classification, we perform a majority vote for the class breaking ties randomly.

The models and training process are implemented in python using the PyTorch [390] and Fastai [391] libraries. We use PyTorch to build the architecture, and Fastai to find an appropriate learning rate and handle the training loop with a one-cycle policy [392, 393] and an Adam optimizer [394]. As mentioned before, the code can be found in the repository from Ref. [380].

7.3 Results

As we have mentioned earlier, the various anomalous diffusion models exhibit very distinguished characteristics. Therefore, we first address the trajectory classification problem (Task 2 of the challenge) using our HYDRA model. This task is evaluated in terms of the F_1-score metric, computed as a function of the true positives (TP), false positives (FP), and false negatives (FN):

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}\,. \tag{7.1}$$

We train an ensemble of ten bi-headed HYDRAs for the task that achieved an F_1-score of 0.83 in the last phase of the challenge.

Subsequently, we tackle the regression of α (Task 1 in the challenge), which is evaluated with the mean absolute error (MAE):

$$\mathrm{MAE} = |\alpha_{\mathrm{true}} - \alpha_{\mathrm{pred}}|\,. \tag{7.2}$$

Given that every anomalous diffusion model deviates from normal diffusion in a different way, their respective α values are associated to different specific phenomena for every model. For instance, α is related to the waiting time distribution of CTRWs, but it dictates the power-law dependence of the diffusion coefficient in SBM trajectories. For this reason, we train a specialized bi-headed HYDRA to infer α for each anomalous diffusion model, resulting in five HYDRAs. Then, we combine the specialized HYDRAs with the classifier from Task 2 in a unified HYDRA with 12 heads: 2×5 experts plus 2 from the classifier. With this design, a final fully-connected block incorporates the opinion of each expert along with the classifier output, which assesses which expert should be trusted, as illustrated in Fig. 7.1(b). Following this strategy, we build an ensemble of ten 12-headed HYDRAs that achieved a MAE of 0.29 in the final phase of the competition.

We refer to Ref. [367] for a comprehensive analysis of the overall AnDi challenge results and alternative proposals to study anomalous diffusion.

Chapter 8

STEP: extracting pointwise diffusion properties

In this chapter, we explore how machine learning (ML) techniques can be used to study diffusion processes that feature time-dependent properties. In particular, we will demonstrate the utility of a ML method, STEP, for the study of changes in anomalous diffusion, and its emergence from changes in normal diffusion. We begin by introducing the method and validating it using simulated data. Then, we proceed to apply it to investigate the diffusion properties of membrane proteins in cells.

This chapter is based on the work we presented in Ref. [389]. We provide an accompanying library in Ref. [395], containing the code and extensive tutorials to reproduce the results.

8.1 Changes of diffusion

Advances in optical imaging enable the direct observation of single molecules in living biological systems [396]. By combining these imaging techniques with particle tracking algorithms, it becomes possible to trace the movement of individual elements with nanometric precision, from molecules and viruses to organelles, enabling the study of transport mechanisms in complex biological environments. As we have presented in Chapter 6, the biophysical characterization of these trajectories allows us to extract meaningful parameters to describe physical and biological processes. However, accurately quantifying the trajectories remains a challenge [352].

Nowadays, there exists a plethora of methods to characterize diffusive processes. As discussed in Sections 6.1 and 6.3, even though the techniques based on the estimation of the mean squared displacement (MSD) may result in optimal estimators for the diffusion coefficient D in certain scenarios [353, 354], they suffer from major limitations in the study of anomalous diffusion. Even the alternative methods presented throughout Section 6.3 and Chapter 7, which are based in extracting meaningful information from the trajectories, performing Bayesian estimation, or using ML algorithms, are mostly limited to study trajectories with constant diffusive properties.

In cellular systems, time-dependent changes of motion are a prevalent aspect of diffusion [397]. Typically, these changes are associated with transient interactions with other components [398–400] resulting in the sudden variation of a parameter, such as D, that may switch between a discrete [401], or continuous set of levels [324, 351, 402]. Furthermore, they can induce smooth changes such as those associated with the spatio-temporal heterogeneity of the environment [348]. Examples of trajectories undergoing changes of diffusion are schematically depicted in Fig. 8.1A.

Trajectories with time-dependent diffusion properties pose an additional challenge to characterize the motion of individual particles, which has been tackled with

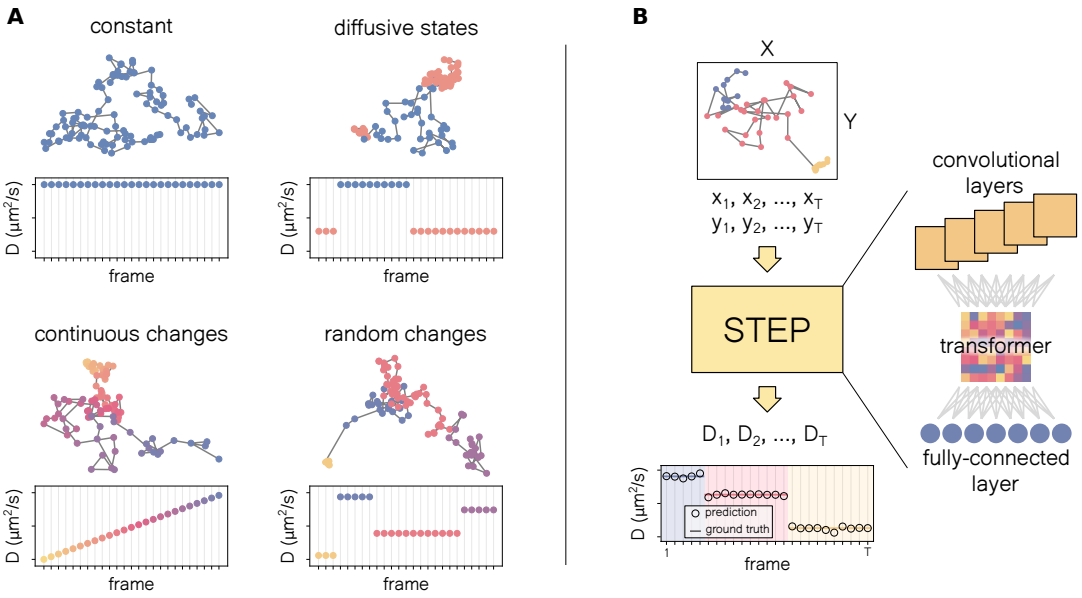

FIGURE 8.1: **Heterogeneous trajectories and the STEP pipeline.** (**A**) Examples of trajectories and the corresponding diffusion coefficient D as a function of time for: constant D; changes within a discrete set of states with fixed D; continuous and monotonous change of D; switch between random Ds. (**B**) Schematic of the pipeline of STEP: an input trajectory is fed to the architecture, which consists of a stack of convolutional layers, a transformer encoder, and a pointwise feedforward layer. The model's output is the pointwise prediction of the diffusion parameter of interest (in this case D).

different approaches. For trajectories displaying abrupt changes, the combination of statistical methods with segmentation algorithms [397, 403, 404] are a valuable strategy but cannot deal with long-range correlations and often offer limited time resolution due to temporal averaging. On the other hand, model-dependent methods such as the hidden Markov model have been quite successful in describing heterogeneous diffusion [405–407], although they require prior knowledge about the diffusive states involved and their kinetic scheme. Recently, data-driven approaches have shown remarkable capabilities to extract information from individual stochastic trajectories, even in the presence of changes of diffusion properties [374, 378, 408, 409].

In Ref. [389], we propose STEP, a method based on state-of-the-art deep learning (DL) architectures to extract pointwise diffusion features from individual trajectories without any prior information (see Fig. 8.1B). This allows STEP to overcome many of the presented methods' limitations, making it suitable for a wide range of applications. As we detail in the upcoming sections, STEP features the most recent advances in sequence-to-sequence learning [410], which have shown impressive results in natural language processing tasks and beyond [67, 411, 412], and allow STEP to achieve remarkable performance in a wide range of scenarios.

8.2 Methods

8.2.1 The STEP architecture

Recently, we have witnessed an enormous effort in the development of DL approaches to study diffusive processes, as we have introduced in Section 6.3. Previous works usually focused on characterizing diffusive properties of single trajectories by

predicting an overall or average diffusive parameter for each of them [363–366, 375], which has already proven to be a valid approach to study complex phenomena in some experimental scenarios [373, 374].

With STEP, we propose a sequence-to-sequence approach [410] that translates position coordinates into the diffusion properties of interest at every time step of the input trajectory, as illustrated in Fig. 8.1B. In this way, the input and the output of the model have the same length. While it is effectively impossible to characterize diffusion from a single displacement due to its stochastic nature, STEP uses the whole trajectory as context to perform the prediction at every point.

This approach allows us to study trajectories whose diffusion properties can vary over time with different patterns: from trajectories with constant diffusive properties to trajectories that sharply switch between different diffusive states, or with diffusive parameters that change continuously over time (see Fig. 8.1A for examples). Unlike previous works, where expert input is needed in order to choose an appropriate method, STEP can be seamlessly applied to any diffusive data. Importantly, it does not rely on prior assumptions, such as the number of changepoints [368, 369] or the properties of the expected diffusive states [413].

In diffusion phenomena, we deal with complex statistical signals that can exhibit various types of time correlations. Furthermore, we often encounter trajectories with very different lengths, even in the same experiment. Hence, it is crucial that the ML models are *length-independent* and able to capture correlations at different time scales to ensure that they are as applicable as possible. In Section 6.3, we have seen that many state-of-the-art architectures for diffusion characterization rely on very different approaches, although combinations of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are the most prevalent. We propose an architecture for STEP combining CNNs with self-attention mechanisms. Interestingly, similar strategies have been adopted in recent works for novel supervised [378, 388] and unsupervised [377] methods.

First, the input trajectory is processed by a series of convolutional layers that we build following the XResNet architecture [61]. This expands the trajectory dimension to provide a richer representation based on local features. Then, the result follows through a transformer encoder [66], which can capture global correlations. Finally, we use a pointwise fully-connected layer of non-linear neurons to obtain the desired output dimension containing the diffusion properties, as we illustrate in Fig. 8.1B. We refer to Section 2.2.2 for details about the different blocks, and to Appendix A for technical details about the proposed architecture, such as the number of layers and their sizes.

8.2.2 Training procedure

We train two models: one to infer the diffusion coefficient D, and another for the anomalous diffusion exponent α. Each model is trained on simulated noisy trajectories that feature abrupt changes in their respective diffusion properties (D or α), and the goal is to predict those properties at every time step of the input trajectories. In the following Section 8.2.3, we provide details about the data used to train and validate the models.

We follow a standard gradient-based training procedure for both of them, and the only differences arise from the training data and how we process it. The main training loop consists on:

i. Predict the values over a batch of training data.

ii. Compute the loss function with respect to the true values.

iii. Update the model parameters based on the loss gradient.

We use batches containing 128 trajectories and the L_1 loss function, which corresponds to the mean absolute error (MAE). Formally,

$$\mathcal{L}_{\text{MAE}}(\boldsymbol{x}) = \frac{1}{N\sum_i T_i}\sum_{i=1}^{N}\sum_{t=1}^{T_i}|y_{i,t} - f(\boldsymbol{x}_i)_t| \ , \tag{8.1}$$

where $f(\boldsymbol{x}_i)_t$ denotes the prediction of the i-th trajectory at the time step t in a batch of N trajectories. $y_{i,t}$ denotes its true label for the same time and trajectory, and T_i denotes its total length.

The whole procedure is implemented using the PyTorch [390] and Fastai [391] python libraries. To perform the parameter update, we use an Adam [394] optimizer. We choose the learning rate with the learning rate finder tool of the Fastai library, which typically is of the order of 10^{-4}. Then, we implement a schedule over the training batches both in the learning rate and its momentum, following the one-cycle policy introduced in Refs. [392, 393]. We train our models until the performance in the validation set stabilizes, typically between ten to twenty epochs.

To further prevent overfitting and enhance the model's generalization capabilities, we use dropout [50] and weight decay [46, 47]. Additionally, we add Gaussian localization noise at different intensities to the trajectories as a form of data augmentation to make the models robust to noise characteristic of experimental settings.

8.2.3 Data

We generate multiple data sets in order to train and properly evaluate STEP. We use a dedicated data set to train each of the models, one to infer D and the other to infer α, and the rest are used to test them on unseen scenarios, which we design to evaluate different aspects of the models.

In Appendix A, we provide the details about the data sets that we use to train, validate and test our models. These data sets contain simulated trajectories with their corresponding labels at each time step. We have two main approaches to simulate the trajectories depending on whether we deal with normal or anomalous diffusion. Below, we explain how we generate the data for both cases. However, there are common factors for both approaches: the trajectories have piecewise constant diffusion properties, every segment has a minimum length of 10-time steps, and all the trajectories are 2-dimensional.

Brownian motion – We simulate Brownian motion trajectories by taking uncorrelated Gaussian noise as the trajectory displacements. We control the diffusion coefficient at each time step with the standard deviation of the Gaussian noise, which corresponds to $\sqrt{2D}$. This way, we can easily generate segments of arbitrary lengths with a constant diffusion coefficient, D, along the trajectories. Finally, we perform the cumulative sum of the displacements to obtain the trajectory coordinates and we subtract the initial position such that they start at the origin.

We consider diffusion coefficients across six orders of magnitude $D \in [10^{-3}, 10^{3}]$. However, we take its logarithm as labels for the regression task, such that $y_i \in [-3, 3]$ at every time step. This greatly simplifies the problem and allows us to keep a consistent performance across all orders of magnitude.

Additionally, we can simulate experimental localization noise by adding Gaussian noise with standard deviation σ_{noise}. We use this as a form of data augmentation

during training and to study the model's resilience to noise. See Table A.1 for further details.

Anomalous diffusion – To simulate anomalous diffusion trajectories, we consider the five diffusion models introduced in Section 6.2. We generate full trajectories for each model following the same procedure detailed in the Supplementary Material from Ref. [367] and using the `andi_datasets` python library [381]. Then, in order to obtain heterogeneous trajectories, we split them into segments, shuffle them, and recombine combine them together. We impose the condition that two consecutive segments must differ, at least, either in the diffusion model or α. Then, we add Gaussian localization noise, with standard deviation σ_{noise}. Finally, we normalize the resulting displacements by their standard deviation and subtract the initial position to ensure that the trajectory starts at the origin.

Therefore, we have two labels at each time step: the anomalous diffusion exponent and α with which the corresponding segment was generated. This allows us to use the same data for both a regression task in the α and a classification task in the diffusion model. However, we mainly focus on the first one. Furthermore, we balance all the data sets such that there is an even representation of both the anomalous diffusion exponents and diffusion models throughout all the time steps.

8.2.4 Baselines

STEP provides pointwise diffusion properties for input trajectories, which, to the best of our knowledge, is a distinct task from that addressed by any existing method. In this context, it is challenging to perform a straightforward and equitable comparison of STEP with these methods, particularly considering that typical approaches involve the combination of various methods to achieve similar results.

Thus, instead of creating complex benchmarks, we compare STEP to the best alternatives proposed so far for the extraction of diffusion properties such as D and α. Since these methods cannot deal with time-dependent changeds of diffusion, we apply them to trajectories pre-segmented according to the ground truth. For calculating D from Brownian trajectories, we employ the fitting of the time-averaged MSD (TA-MSD), which is the optimal estimator for D in most cases [353]. For estimating α from trajectories undergoing anomalous diffusion, we employ the TA-MSD fit in logarithmic space and leverage CONDOR [368], recognized as the leading approach for this task in the AnDi Challenge [367]. While these methods benefit of a significant advantage due to the pre-segmentation, STEP typically achieves comparable performance (see Fig. 8.2).

8.3 Results

8.3.1 Pointwise prediction of diffusion properties

We first validate STEP on the task of inferring the pointwise diffusion coefficient from simulated trajectories reproducing transient Brownian motion with abrupt changes of diffusion coefficient. The diffusion coefficient can randomly vary in the range $D \in [10^{-3}, 10^{3}]$ and the dwell time in each diffusion coefficient is drawn from an exponential distribution between 10 and 190 (mean 57) time steps. A 2D histogram of the ground truth versus the predicted diffusion coefficient shows that STEP can precisely determine the diffusion coefficient across its whole range (Fig. 8.2A) with an overall relative error $|D_{\text{true}} - D_{\text{pred}}|/D_{\text{true}} = 0.226$.

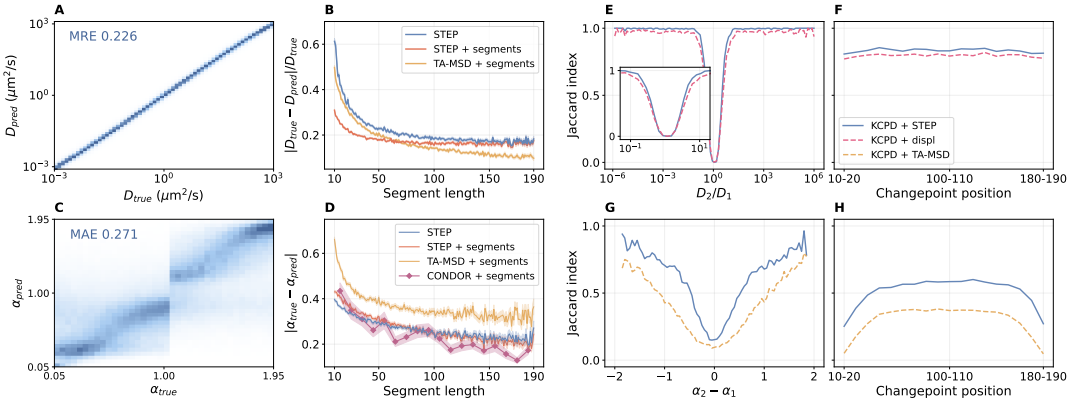

FIGURE 8.2: **Time-dependent diffusion properties prediction.** (**A**) 2D histogram of the predicted diffusion coefficient D compared to the ground truth. The mean relative error (MRE) over the whole test set is 0.226. (**B**) Relative error for the prediction of D as a function of the segment length. (**C**) 2D histogram of the predicted anomalous diffusion exponent α compared to the ground truth. The MAE over the whole test set is 0.271. (**D**) MAE for the prediction of α as function of the segment length. (**E** to **H**) Jaccard index (JI) for the changepoint detection problem as a function of: (**E**) the ratio between consecutive segment Ds , (**F**) the changepoint position for D, (**G**) the difference between consecutive segment αs, and (**H**) the changepoint position for α. For details about the data used in each panel, see Table A.1.

To further explore the performance of STEP, we calculate the relative error as a function of the segment length, i.e., the dwell time for each D (blue line in Fig. 8.2B). The error increases for shorter segments, meaning that STEP needs sufficient statistics from surrounding points with similar properties to accurately predict the diffusive properties of a given point. Comparing these results with the TA-MSD's prediction on pre-segmented trajectories, STEP is close to the optimal target performance (blue vs yellow lines in Fig. 8.2B). Notably, when STEP is provided with pre-segmented trajectories, we observe a further improvement, with a nearly 2-fold reduction of the error at short segment lengths (red line in Fig. 8.2B), demonstrating outstanding prediction capabilities. However, it is outperformed by the TA-MSD baseline in long segments.

We take closer look at this performance trade-off between STEP and the TA-MSD estimator of D, which is the optimal estimator in these conditions [353]. We assess the prediction uncertainty of the different methods as a function of the segment length and compare it to the Cramér-Rao lower bound (CRLB), as shown in Fig. 8.3(a). The TA-MSD fit is, indeed, the optimal unbiased estimator for D, as it aligns perfectly with the CRLB (yellow and dashed black lines). Conversely, STEP consistently violates the CRLB (blue and red lines below the dashed black line), suggesting a degree of bias in its estimation.

Several factors can contribute to introducing bias in the resulting model. For example, STEP estimates the logarithm of D, which typically leads to lower variance estimators at the cost of introducing a potential bias. It is also essential to note that the resulting model heavily depends on both the training data and the loss function employed during training. In this specific case, the distribution of segment lengths in the training data set follows an exponential function, resulting in a higher proportion of shorter segments. From the perspective of minimizing the loss, there is a tendency to prioritize the performance for the shorter segments, even if it comes at the expense of the longer ones.

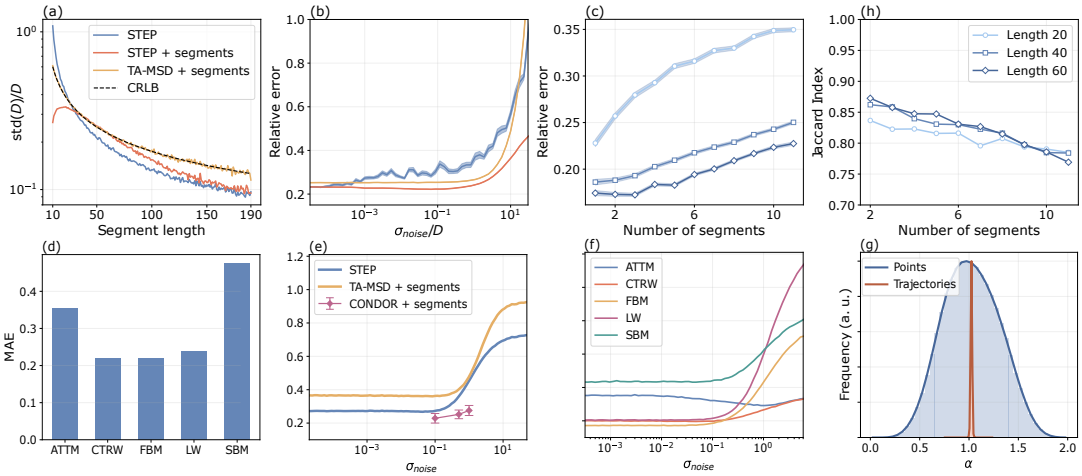

FIGURE 8.3: **STEP performance deep dive for D (top row) and α (bottom row)** (a) Prediction uncertainty for D comparing all the methods to the CRLB. (b) Relative error in D as a function of the ratio between the localization noise's standard deviation σ_{noise} and D. (c) Relative error of STEP as a function of the number of segments at three different segment lengths. (d) Prediction MAE by diffusion model. (e) MAE as a function of the localization noise for STEP, TA-MSD, and CONDOR, the last ones with with known segments. (f) MAE as a function of the localization noise separated by diffusion models. The blue line in (e) corresponds to the average of the lines presented in this panel. (g) Prediction of α for Brownian motion trajectories. The blue distribution shows the pointwise prediction for the trajectories, while the red one shows the mean prediction over trajectories. The distributions have been normalized to have the same maximum value. (h) Changepoint detection performance as a function of the number of segments at three different segment lengths. For details about the data used in each panel, see Table A.1.

Then, we consider additional factors that may impact the performance, such as the number of changepoints or the presence of noise. In experimental scenarios, trajectories are affected by localization noise, which is usually modeled as Gaussian noise of variance σ^2_{noise} added to the trajectories. Since we consider diffusion coefficients at very different scales along the trajectories, in Fig. 8.3(b), we plot the error as a function of the ratio between the noise's standard deviation and the diffusion coefficient. We see that STEP strongly outperforms the TA-MSD approach with known segments even well beyond the noise levels present in relevant experimental scenarios (usually $\sigma_{\text{noise}}/D < 10^{-1}$). However, the presence of noise increases the difficulty of accurately finding the boundaries between segments, thus hindering the overall performance of STEP (blue line).

Finally, in order to investigate the effect of the number of changes on the characterization of the trajectories, we fix the segment length and generate trajectories with one to eleven segments (zero to ten changepoints), resulting in trajectories with very different lengths (see Appendix A for details). In Fig. 8.3(c), we see a slight increase of the relative error with the number of segments, although it has a much lesser impact than the segment length. For instance, it is harder to characterize a single segment of twenty points than eight consecutive segments of 40 time steps each. Importantly, even in the presence of 10 changepoints, STEP still heavily outperforms the TA-MSD approach applied to segments (no changes) of the same size, e.g., the whole curve for a segment length of 20 in Fig. 8.3(c) is well below the TA-MSD point for a segment length of 20 in Fig. 8.2(b).

We then examine the ability of STEP to predict the anomalous diffusion exponent

α. We consider trajectories composed of segments following the same length distribution as in the Brownian motion case. Each segment is simulated using a different anomalous diffusion model and $\alpha \in [0.05, 2]$. The 2D histogram of the ground truth vs the predicted α in Fig. 8.2(c) shows that STEP successfully predicts the anomalous diffusion exponent. We obtain a MAE $|\alpha_{\text{true}} - \alpha_{\text{pred}}| = 0.271$, which is in line with the top-scoring approaches for this task [369, 370] in the AnDi challenge [367]. As most methods in the challenge, STEP is prone to errors for $\alpha \sim 1$. In addition, since several models are inherently only sub- or super-diffusive and the method tends to predict values of α within the training range, we observe a discontinuous behavior in the histogram for $\alpha \sim 1$.

In Fig. 8.2(d), we report the MAE for α as a function of the segment length. STEP strongly outperforms the TA-MSD approach and shows a performance comparable to CONDOR. For this task, providing pre-segmented data to STEP marginally improves its performance for long segments, whereas it even reduces it for short ones. This result suggests that segment length is more important than the exact knowledge of the segment edges and STEP effectively combines local and global information.

Given the sheer differences between anomalous diffusion models, we evaluate the performance of STEP segregated by diffusion model. We report the MAE over all segments belonging to each anomalous diffusion model in Fig. 8.3(d). We observe clear differences between models, with continuous time random walk (CTRW), fractional Brownian motion (FBM), and Lévy walk (LW) segments holding the lowest errors. The MAE over scaled Brownian motion (SBM) segments is significantly larger than in the other models. This has already been observed in previous works (see for instance Fig. 2d of Ref. [367]), although the differences here are larger. A detailed inspection shows that the biggest errors come from shorter segments, in agreement with the results from Fig. 8.2(d). This is reasonable since the aging in SBM is the source of the anomalous diffusion [334] and therefore it requires longer segments to be correctly characterized. STEP displays a clear tendency to predict $\alpha \approx 0.8$ for SBM segments, which is enhanced by the presence of noise. This behaviour suggests that the model struggles to identify any clear behavior in short segments and defaults to a prediction that minimizes the overall possible errors.

Interestingly, we find a similar trend in CTRW segments, where the model has a tendency to predict $\alpha \simeq 0.25$, corresponding to nearly immobile particles. As we have introduced in Section 6.2.1, CTRW trajectories are characterized by jumps at random times, resulting in segments in which the particle does not move. Hence, many CTRW segments in our heterogeneous trajectories do not display any movement due to their short lengths, corresponding to a waiting time window. Therefore, it is impossible for the model to correctly predict α, as it does not have any information to work with.

To a lesser extent, we also find that the model predicts $\alpha \sim 1$ for low anomalous diffusion exponents in annealed transit-time model (ATTM) segments. In ATTM trajectories with small α, we encounter very long segments with low diffusion coefficients. Similar to the CTRW case, we encounter parts of these long segments in our heterogeneous trajectories containing a unique diffusion coefficient. Thus, they effectively behave like Brownian motion along the observed time window, justifying the predictions $\alpha \sim 1$ in these cases.

We proceed to study the resilience of the method to localization noise, as we have done for the case of D. We present the MAE as a function of the noise's standard deviation σ_{noise} in Fig. 8.3(e). We observe a consistent performance of all the methods until reaching considerable levels of noise. Again, STEP is comparable to CONDOR despite the latter having the advantage of knowing the segments beforehand. In

Fig. 8.3(f), we show the impact of localization noise for each diffusion model. As we have seen throughout this section, characterizing some diffusion models is harder than others, and the localization noise has a different impact on them. While increasing the noise level has an overall negative effect, the performance on LW segments suffers the most, while the performance on CTRW segments is barely affected. Overall, the errors start to increase significantly beyond $\sigma_{\text{noise}} \sim 2 \times 10^{-1}$, which would correspond to harsh experimental conditions. Interestingly, ATTM segments see a drop in MAE with increasing noise for a limited range.

To conclude the performance analysis, we use the model trained to characterize anomalous diffusion to predict α for the Brownian motion trajectories with random diffusion changes from Fig. 8.2(a) & (b). We show the results in Fig. 8.3(g). The pointwise prediction (blue distribution) averages to $\alpha = 1.02$, which aligns closely with the expected value of 1. This outcome gains further credibility when we compute the average predicted α across the entire trajectory (red distribution), since it shows that all trajectories are consistently predicted to have $\alpha \approx 1$. Although these trajectories closely resemble ATTM trajectories, characterized by random changes in D, STEP correctly discerns that the trajectories in question do not exhibit anomalous diffusion. This determination is based on the fact that the distribution of D and dwell times τ do not satisfy the conditions outlined in Section 6.2.3, which are necessary to induce anomalous behavior.

8.3.2 Detecting diffusive changepoints in heterogeneous trajectories

For trajectories undergoing sudden changes of diffusion properties, the exact knowledge of the points at which these changes occur is crucial to infer temporal properties and kinetic rates of the system and fully characterizing the underlying physical process. While STEP does not explicitly detect changepoints, its output provides a precise estimation of the diffusion property which is supposed to change, hence simplifying the task of changepoint detection and location with respect to the use of raw data. To highlight this capability, we compare the results obtained by a state-of-the-art kernel changepoint detection (KCPD) method [414, 415] when applied to STEP's predictions and to the time series of trajectory displacements. We use the *ruptures* Python library [416] for the KCPD implementation. To assess the performance of the methods, we compute the Jaccard index (JI) considering as a true positive any changepoint predictions lying within a threshold distance $\mathcal{E}$ from the corresponding ground truth. The JI is computed as a function of the true positives (TP), false positives (FP), and false negatives (FN):

$$\text{JI} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} . \tag{8.2}$$

We first quantify the performance of the method to detect changes of the diffusion coefficient in heterogeneous Brownian motion trajectories. We use a benchmark data set with trajectories of 200 time steps exhibiting a single changepoint and set $\mathcal{E} = 5$. Applying the KCPD algorithm on the prediction of STEP, we can successfully detect the changepoints with high accuracy. The detection improves as the differences between consecutive segments increase, achieving a nearly-perfect detection for segments whose diffusion coefficients are just one order of magnitude apart, as we show in Fig. 8.2E (blue line). Furthermore, the method is robust with respect to the changepoint position within the trace (Fig. 8.2(f) (blue line)). In contrast, when we apply KCPD directly over the trajectory displacements (dashed purple lines) we

observe a decrease in performance over the whole range of diffusion coefficient ratio. On average, STEP produces a 20% reduction in error reaching an average JI of 0.833 compared to 0.796 obtained with the raw displacements.

Similar to Section 8.3.1, we also evaluate the performance as a function of the number of segments, depicted in Fig. 8.3(h). Detecting changes between short segments is much harder than between longer ones. However, segment length becomes less important as they become longer, as the curves for lengths 40 and 60 behave fairly similarly. STEP achieves a better score for shorter segments when the trajectories are very long (11 segments). This suggests that every additional changepoint in the trajectory adds a similar amount of error sources which are eventually outweighed by the accumulated errors along the trajectory as it gets longer. Nonetheless, even in the most challenging cases, STEP correctly detects the vast majority of the points.

We perform a similar analysis to detect changes in the anomalous diffusion exponent. To ease the analysis, we consider only trajectories undergoing FBM [343] (see Section 6.2.4 for details). Since the anomalous diffusion exponent is an asymptotic property that cannot be easily calculated from the raw data, to build our baseline, we compute α with a linear fit of the TA-MSD on a log-log scale using a sliding window of 30 time steps, which we then feed into the KCPD algorithm (dashed yellow lines). Expectedly, the larger the differences between segment parameters, the better we can detect the changepoints, as we show in Fig. 8.2G. We obtain a 30% error reduction by using STEP with respect to the baseline method, achieving an average JI of 0.515 and 0.297, respectively, with $\mathcal{E} = 20$. However, these metrics show that finding changes in α is significantly harder than in D. Moreover, we also observe a performance drop when the changepoints are near the trajectory edges (Fig. 8.2H). In these cases, we deal with short segments whose anomalous diffusion exponent can be hard to determine, as they rely on the arising of long-range correlations.

8.3.3 Revealing continuous changes of diffusion properties

When considering heterogeneous trajectories in the biological context, the typical behavior one expects is represented by particles undergoing diffusion with piecewise constant properties that can suddenly change, e.g., as the result of specific interactions with other biological components. However, the presence of molecular crowding and gradients of concentration can produce a continuous variation of diffusion properties over time [351]. These changes might be challenging to detect due to the limited spatio-temporal resolution of the experiments or the lack of specific approaches for trajectory analysis. Since STEP predicts pointwise diffusion properties in a model-free fashion, it inherently features the capability to perform this kind of analysis, even without dedicated training.

To evaluate the performance of STEP on smoothly-varying trajectories, we rely on simulations of SBM [348]. As detailed in Section 6.2.5, SBM trajectories are characterized by a time-dependent diffusion coefficient with a power-law relationship $D(t) \sim t^{\alpha-1}$.

In Fig. 8.4, we show the predictions obtained for the diffusion coefficient at every time step of trajectories with $\alpha = 0.1$ and 0.5. The shaded lines represent the STEP predictions obtained for individual trajectories which, despite the fluctuations, already indicate the decreasing trend. Averaging over trajectories (round marker) reveals the correct power-law scaling (dashed lines). We also obtain the correct scaling with a linear fit of the TA-MSD on a sliding window of 20 points over the trajectories (solid lines). As shown, STEP can capture the scaling earlier, since it is not limited by

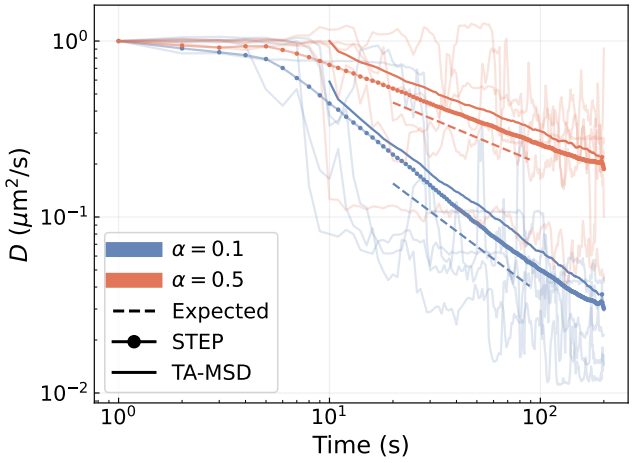

FIGURE 8.4: **Continuous changes of diffusion properties.** Predictions of the time-dependent diffusion coefficient of two sets of SBM trajectories with $\alpha = 0.1$ (blue) and 0.5 (red). The bold continuous lines show the average prediction over 3000 trajectories with STEP and a linear fit of the TA-MSD over a sliding window of 20 points. The thin continuous lines show a few example STEP predictions. The dashed lines indicate the theoretically expected scaling for every α. The lines have been normalized and shifted to compare their slopes easily.

the size of the window. Furthermore, the inference of α correctly provides a nearly constant value throughout the trajectory, as expected.

8.3.4 Characterizing anomalous diffusion from changes of normal diffusive properties

To test the potential of STEP for the analysis of experimental trajectories, we use it to study the motion of the pathogen-recognition receptor DC-SIGN expressed in Chinese hamster ovarian cells [417]. Previous analysis of these experiments revealed the occurrence of anomalous diffusion and weak ergodicity breaking as a consequence of stochastic changes of diffusion coefficient [324]. This behavior was described in the framework of the ATTM [341], whose main features are schematically summarized in Fig. 8.5A.

As we have introduced in Section 6.2.3, ATTM depicts Brownian diffusion with random piecewise constant D, whose segment lengths τ depend on the value of D (Fig. 8.5A, top). This results in anomalous and weakly non-ergodic diffusion, despite performing Brownian motion at short scales. In the regime of ATTM we are interested in, D is sampled from a probability distribution with a power-law behaviour $P(D) \sim D^{\sigma-1}$ for small D and a fast decay for $D \to \infty$ (Fig. 8.5A, middle). Moreover, the dwell time τ is correlated to D and it is sampled from a distribution of the form $\mathbb{E}[P(\tau|D)] = D^{-\gamma}$ (inset of Fig. 8.5A, middle), resulting in an effective $\alpha = \sigma/\gamma$ (Fig. 8.5A, bottom). Therefore, the correct characterization of both the distribution of $P(D)$ and $P(\tau|D)$ is crucial to corroborate the compatibility with the underlying model. In the original work [324], changes of diffusivity were detected through a changepoint analysis [418] but the sensitivity and the time-resolution of the method did not allow a thorough investigation of this behavior.

To demonstrate that STEP enables a better characterization of this data, we first use simulated ATTM trajectories. We set $\sigma = 0.3$ and $\gamma = 0.4$ ($\alpha = 0.75$), resulting in trajectories with $D \in (10^{-6.7}, 10^0)$ with 18 different segments, on average, for

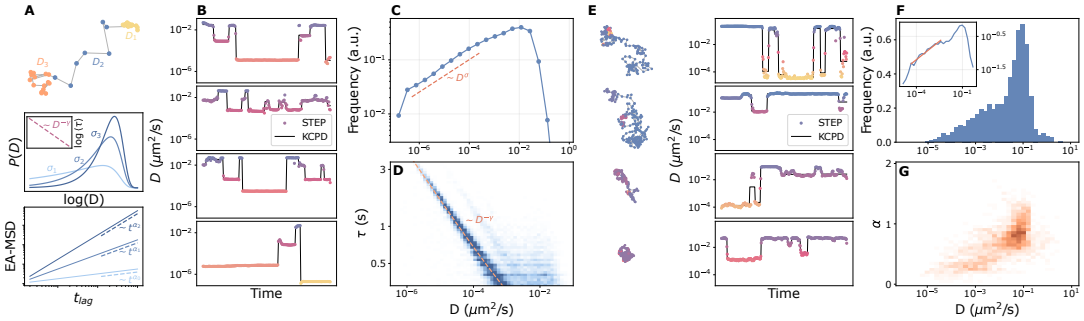

FIGURE 8.5: **Switch between random diffusive states of the pathogen-recognition receptor DC-SIGN.** (**A**) Characteristic features of the ATTM model: an exemplary trajectory undergoing changes of diffusion coefficient; a few examples of the distribution of D with different σ_i, and an example relation between the diffusion coefficient and the dwell time τ for a fixed γ; the ensemble-average MSD scaling for the $\alpha_i = \sigma_i/\gamma$ that result from each of the previous σ_i and a fixed γ. (**B**) Predictions of the diffusion coefficient obtained by applying STEP to simulated ATTM trajectories (dots) and the result of applying the changepoint analysis (black line). (**C**) Distribution of D obtained through the analysis described in (**B**), showing the expected power law behavior at small D. (**D**) Relation between D and the dwell time τ obtained through the analysis described in (**B**), showing the expected power law behavior. (**E**) Examples of experimental trajectories of DC-SIGN with the corresponding predictions obtained for D (dots) and the changepoint analysis (black line). (**F**) Histogram of the distribution of D obtained for the experimental trajectories. Inset: power-law fit at small D. (**G**) 2D histogram of D and α obtained for the experimental trajectories. For details about the data used in each panel, see Table A.1.

trajectories of 200 time steps. We segment the trajectories applying the KCPD algorithm introduced in the previous sections over the STEP predictions of D, as we show in Fig. 8.5B. Thus, we assign to each segment a single D, taking the average segment prediction, and a τ. We successfully recover the power-law behavior of D (Fig. 8.5C) and the power-law relationship between τ and D (Fig. 8.5D). The faint harmonic in Fig. 8.5D corresponds to $2D^{-\gamma}$, which results from the missed detection of a changepoint between consecutive segments with very similar D (hence similar τ). Interestingly, when performing predictions of α, STEP predicts $\alpha \sim 1$, as expected from the properties of the diffusion model (see **??**).

Then, we apply this approach to the DC-SIGN trajectories of Ref. [324]. The results confirm the occurrence of diffusivity changes between segments of nearly-constant diffusion coefficient and with variable duration, as we show in Fig. 8.5E. Interestingly, our approach reveals twice as many changepoints as the previous analysis.

The distribution of D obtained for trajectory segments spans several orders of magnitude, as we show in the histogram of Fig. 8.5F. For small D, it displays a behavior compatible with a power-law with exponent $\sigma \approx 0.37$ over nearly three decades (inset of Fig. 8.5F), compatible with the ATTM. Notably, this behavior could not be directly verified in the original article. In principle, our method would allow us to verify the correlation between D and dwell time, as we have shown in the simulations. However, this task is limited by the variable trajectory length [419] and by the lack of statistics, in particular for segments at small D.

As a further test, we predict the anomalous diffusion exponent with STEP. We assign a single α by taking the average prediction of each segment. The results reported in Fig. 8.5G show an interesting correlation between D and α that suggests a more complex diffusion pattern, involving the occurrence of anomalous diffusion also at the level of individual segments.

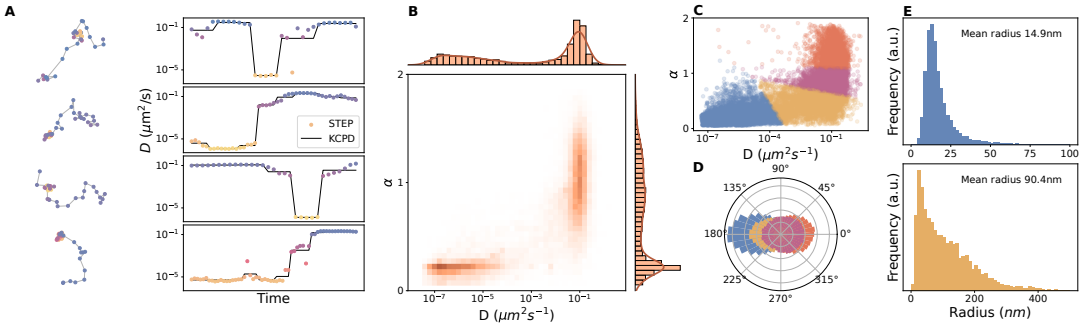

FIGURE 8.6: **Multi-state diffusion of the integrin $\alpha 5\beta 1$.** (**A**) Examples of experimental trajectories of the integrin $\alpha 5\beta 1$ with the corresponding predictions obtained for D (dots) and the changepoint analysis (black line). (**B**) 2D histogram of D and α with the respective marginal distributions. (**C**) Scatter plot of the predictions obtained for D and α at the segment level, color-coded according to a clustering analysis performed with a k-means algorithm. (**D**) Distribution of the turning angle for the four clusters of segments obtained as in (**C**). (**E**) Distribution of the confinement radius for the clusters showing restrained diffusion. For details about the data in each panel, see Table A.1.

8.3.5 Characterizing multi-state diffusion processes

We use STEP to analyze experimental trajectories of the integrin $\alpha 5\beta 1$ diffusing in the membrane of HeLa cells. Integrins are transmembrane receptors for the extracellular matrix (ECM) in focal adhesions, which mechanically link the ECM and actin filaments in the cytoplasm and activate signaling pathways involved in cell migration, proliferation, or apoptosis [420]. The dynamics of the integrin $\alpha 5\beta 1$ is influenced by interactions with fibronectin and actin-binding proteins [419, 421]. Its motion has been reported to switch from fast free-diffusion to slow free-diffusion and immobilization, as well as exhibiting rearward actin-driven movement.

We use STEP to predict both the diffusion coefficient and the anomalous diffusion exponent for the integrin $\alpha 5\beta 1$ trajectories. Then, we segment the trajectories by applying the KCPD method to both predictions at once. In this way, we assign every segment a unique D and α by taking the average prediction over the segment. Examples of the results are shown in Fig. 8.6A and the joint distribution of D and α in Fig. 8.6B. The visual inspection of Fig. 8.6B reveals two main clusters centered around ($D = 10^{-6}\mu\text{m}^2/\text{s}$, $\alpha = 0.25$) and ($D = 0.1\,\mu\text{m}^2/\text{s}$, $\alpha = 1$). The 2D histogram of the same parameters calculated at the pointwise level (pre-segmentation) does not show any major differences with respect to Fig. 8.6B.

Nonetheless, combining the k-means clustering algorithm with the elbow method [422], we find the data optimally separates in four clusters of segments (Fig. 8.6C) characterized by different motion features. The first two clusters show a rather restrained motion, with integrins spending 40% of the time in a state characterized by $D = 1.2 \times 10^{-5}\mu\text{m}^2/\text{s}$ and $\alpha = 0.23$, and 14% of the time with $D = 0.06\,\mu\text{m}^2/\text{s}$ and $\alpha = 0.46$. For both clusters, the distribution of angles between successive steps shows a peak centered at 180°, indicating backward movements due to reflection at potential boundaries, as we show in Fig. 8.6D. The confinement radius of the first cluster has a median of 14.9 nm (st. dev. $\pm$13.3 nm), which is comparable to the localization precision of these experiments. This allows us to associate it with protein immobilization. The second cluster shows confined motion within areas with a broad distribution of sizes, as we see in Fig. 8.6E, and a median radius of 90.4 nm (st. dev. $\pm$98.2 nm). The third cluster represents the 29% of the total recording and shows minor deviations from Brownian motion with $\alpha = 0.88$

and a nearly uniform angle distribution, and has an average $D = 0.10\,\mu\text{m}^2/\text{s}$, close to the value typically reported for this protein. Interestingly, the analysis pinpoints a fourth population, corresponding to a 20% of the total recording, undergoing superdiffusion with $\alpha = 1.3$ and $D = 0.14\,\mu\text{m}^2/\text{s}$, and with a persistent direction of motion between consecutive steps (Fig. 8.6D).

8.4 Conclusion

In Ref. [389], we present STEP, an ML method to predict diffusion properties from individual trajectories at every time step. The method relies on a combination of state-of-the-art DL architectures that take into account correlations at different time scales. The presented approach is especially appealing to analyze trajectories from particles undergoing heterogeneous motion, where changes in diffusion properties occur over time. Moreover, it does not require prior knowledge of the underlying physical process or the temporal resolution at which changes in diffusion occur.

To illustrate the power of STEP, we benchmark it on simulated trajectories under various conditions. We show its ability to predict piecewise constant diffusion properties, such as the diffusion coefficient or the anomalous diffusion exponent, in noisy and short trajectories. Furthermore, we demonstrate that STEP boosts the accuracy of a changepoint detection algorithm to detect the time at which diffusion changes take place. Importantly, we also prove the suitability of our method to study continuous changes of diffusion.

To further demonstrate the potential applications of the method, we study trajectories obtained by tracking live-cell single-molecule imaging experiments of proteins of the plasma membrane. First, we characterize the motion of the pathogen recognition receptor DC-SIGN, which was shown to exhibit random changes in the diffusion coefficient. Our analysis confirms such a hypothesis and improves the accuracy with which we detect these changes. Moreover, our results suggest the occurrence of more complex phenomena that need further investigation. Secondly, we study the diffusion of the integrin $\alpha_5\beta_1$. In agreement with previous works, our analysis confirms the existence of different diffusion modes and allows their precise classification according to the diffusion coefficient, the anomalous diffusion exponent, and the levels of spatial constraint.

We believe that STEP represents a first step towards a new class of ML algorithms to study dynamic systems through a sequence-to-sequence approach. The instantaneous prediction of the property of interest enables the characterization of the trajectories at experimental time resolution without averaging and filtering and minimizes the prior knowledge needed to perform the analysis. As such, the results obtained with STEP can provide information about diffusion properties with unprecedented resolution and thus shed light on the underlying physical processes of a variety of systems. One of the primary advantages of STEP is its broad applicability. However, as demonstrated in the Results section, specialized methods may produce more accurate results when applied specifically to their intended tasks. Consequently, a significant benefit of STEP is its potential integration with these methods, facilitating their utilization across a wider spectrum of scenarios, such as enhancing trajectory segmentation.

Chapter 9

Customer intent prediction

In this chapter, we dive into an industrial application for the study of diffusion with machine learning (ML). Throughout Chapters 6 to 8, we have focused on the biophysical characterization of diffusion trajectories, extracting parameters such as the diffusion coefficient or the anomalous diffusion exponent. Here, we illustrate how similar techniques can be used to study the trajectories described by internet users in their action space in order to predict the outcome of their browsing session. We explore two approaches: combining hand-crafted features with classical ML algorithms, and applying deep learning (DL) models directly to the raw data.

This chapter is based on the work presented in Ref. [423]. The data used for the study is publicly available for research and educational purposes in the repository in Ref. [424].

9.1 The clickstream prediction problem

Upon landing on *exampleshop.com*, a user starts browsing through product listings, raising an intriguing question: will this session culminate in a purchase, or will it conclude without any transaction? The challenge of predicting whether the outcome of a user's browsing session, known as *clickstream prediction* challenge [425–429], is relevant to a broad audience, both from a theoretical and practical perspectives. On the theoretical side, it falls under the umbrella of sequence classification problems [430], akin to those explored in Chapters 7 and 8, the analysis of protein sequences [431], or EEG signal processing [432]. However, the combination of the scale of web inferences, data set characteristics and time constraints make clickstream prediction stand out as a particularly challenging scenario to test statistical tools. On the practical side, the surge in online retail [433], with approximately 25% of all

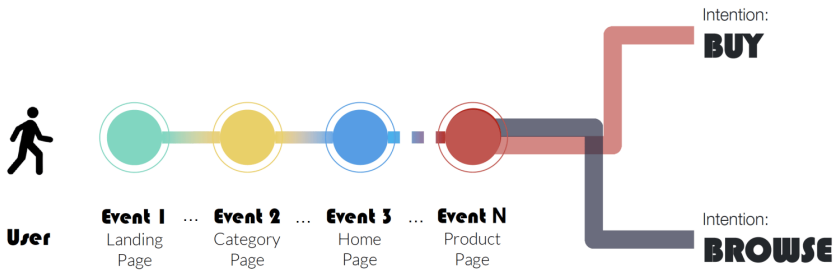

FIGURE 9.1: **Schematic representation of the clickstream prediction problem.** A user reaches an e-commerce landing page and starts browsing the website. Based on its actions, is it possible to asses whether it is just perusing or will it buy a product before leaving?

fashion-related transactions now happening digitally [434], and the connection between browsing behavior and an entire ecosystem of applied science problems [435], makes clickstream prediction a very valuable problem to solve.

A shopper browsing an e-commerce website is analogous to a walker navigating a network enriched with metadata. The shopper generates a trajectory hopping between nodes, with different waiting times between jumps, reminiscent of the phenomena captured by the continuous time random walk (CTRW) model introduced in Section 6.2.1. Therefore, the clickstream prediction problem is analogous to the trajectory characterization tasks addressed in Chapters 6 and 7, where the goal is to assess whether the trajectory is characteristic of a user that purchases an item or not, as depicted in Fig. 9.1. The main difference lies in the nature of the trajectories: while those considered until now traverse the continuous position space, the trajectories described by shoppers unfold over a discrete categorical space.

In Ref. [423], we extensively study the clickstream prediction problem with a novel data set accessible in Ref. [424]. This data set contains rich clickstream data from users browsing a popular fashion e-commerce website. In our study, we consider two main tasks: the classification of clickstream sequences of variable length, and the early prediction of limited-length sequences. To tackle these tasks, we take two different strategies: employing classification algorithms on predefined handcrafted features, and harnessing the power of DL to automatically learn relevant feature representations. The former strategy, being computationally efficient, scalable and interpretable, combines classical symbolic features, such as k-grams, with horizontal visibility graph motifs (HVGMs) [436]. HVGMs are based on the network representation of time series [437], which can be used to study anomalous diffusion [438]. On the other hand, the latter strategy outperforms the former, although it is more sophisticated and less interpretable. In this approach, we extend and enhance state-of-the-art DL models [428] based on recurrent neural networks (RNNs), analogous to the progress in the study of anomalous diffusion [364, 369].

The illustrative example in Fig. 9.2 shows a typical browsing session of a user on an e-commerce website. As the shopper navigates the site, a plethora of data is collected in real-time, forming the basis for clickstream prediction. While the data is rich on metadata, depicted in row **B** of Fig. 9.2, we focus on using minimal information. By removing most of the metadata and symbolizing the trajectories (row **C**), we address a pivotal question: do these symbolized trajectories contain enough information about the user's intent, and can it be exploited and understood? Symbolized trajectories allow us to disentangle the information of clickstream patterns from additional factors, such as waiting times between clicks and the network topological properties of the nodes visited. This approach sets the stage for progressively expanding the research by incorporating more information and evaluating its impact. Furthermore, validating low complexity pipelines is crucial for real-time implementations and decision-making processes in online marketing. Such approaches are less platform-dependent and are, therefore, easier to implement for a broader range of applications.

Notably, previous research on the clickstream prediction problem suffers from low external validity [439]. Experiments on online behavior are often conducted on cases that do not represent the vast landscape of digital shops. For example, methods developed for large-scale shops with exceedingly high conversion rates (e.g., 20% [428]), which is known to vary drastically between countries, industries and even individual shops [440], may not be applicable in more realistic scenarios. In our study, we consider a case with conversion rates below 5%, which serves as an example of a more representative and commonplace situation (typically between 1

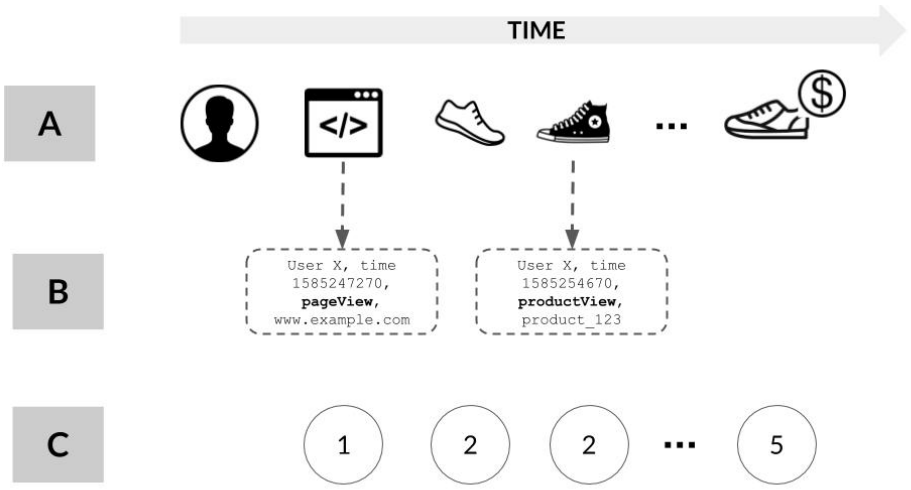

FIGURE 9.2: **Example session and high-level view of the data.** We can distinguish three layers of representation: **A** is the actual session that is being monitored; **B** is the rich meta-data layer, which comprises, for each interaction between the user and the website, information about time, location, product properties, etc.; **C** is a symbolized, minimal information layer, in which all meta-data about the products is stripped and only event types are retained.

to 5% [440]). This huge disparity not only makes class imbalance worse in realistic cases, but also challenges the widespread validity of results obtained on privileged and unusual data sets.

9.2 Data

9.2.1 Raw database

The raw data is provided by Coveo, a North American company providing artificial intelligence (AI) services to the retail and service industries, and it can be accessed through Ref. [424]. The raw database consists of browsing data retrieved from a popular fashion e-commerce website over the course of a two-month period in 2018. The data is stored and anonymized by removing any individual identifier and accessory metadata, such that it is impossible to match events to neither specific products nor individual users. Notably, for this study, users get assigned random identifiers in every session, so it is not even possible to re-identify a user as the same across different sessions.

During the two-month period, all events in the e-commerce website undergo a"sessionization" phase. Every new user is assigned a *session* with the first interaction in the website that lasts a maximum of 30 minutes, which is the industry standard. All the events produced by the same user during that time span are recorded in the same session. Any new event produced by the same user happening after the session time has expired, triggers a new session where subsequent events are recorded. After sessionization, the trajectories are randomly sampled from the resulting data set to protect the data provider from revealing the total amount of traffic on the target website. As a result, the final data set comprises 443,652 anonymized clickstream trajectories of real customers. The next pre-processing step, symbolization, strips all but the most basic event information.

Action	Symbol	Description
Page view	1	Visit any page
Detail	2	Open product page
Add	3	Add product to cart
Remove	4	Remove product from cart
Purchase	5	Purchase items in cart
Click	6	Select a search query result

TABLE 9.1: **Event symbol assignment rule.** Trajectories rich in metadata are mapped into a simple symbolic sequence according to the kind of actions taken by the users.

9.2.2 Trajectory symbolization, class definition, and trimming

The raw data containing the user trajectories is rich in metadata. However, as we have previously introduced, our focus is on using the minimal possible information to perform the clickstream prediction task. In order to minimize the amount of information accompanying each trajectory, we apply a symbolization procedure in which each event is assigned a category according to the action performed by the user that triggered it:

`Page view`**:** when the final user loads any non-product specific page in the website. For instance, the home page or different product listings.

`Detail`**:** when the user visits a specific product page. This involves actions such as navigating to a product's main page, or accessing its details from other pages.

`Click`**:** when the user clicks on a specific result from a search query. For example, searching "shoes" and clicking on one of the returned options.

`Add/Remove`**:** when a product is added/removed to/from the cart.

`Purchase`**:** when the user buys the products in the cart.

Every action category is assigned a symbol according to Table 9.1.

Hence, the problem of clickstream prediction in a session of L actions translates into predicting the appearance of symbol 5 in the symbolized trajectories $\mathcal{S} = (s_1, s_2, \ldots, s_L)$, where $s_i \in \{1, 2, \ldots, 6\}$. Therefore, each trajectory is assigned to one of two possible classes: the *conversion* class (C) and the *non-conversion* class (NC). The conversion class C is assigned to all the trajectories which, at any given point, contain a `Purchase` event, that is, trajectories where the symbol 5 appears at least once. Then, the NC class encompasses all the other trajectories, generated by customers that navigate the website for some time and abandon the session before purchasing anything. In the raw data set, out of the 443, 652 trajectories, 9212 (2.08%) belong to class C and 434, 440 (97.92%) belong to the class NC.

All the class C trajectories are trimmed in order to remove the label information from the observation, keeping only the initial part of the trajectory that precedes the first appearance of a `Purchase` event. As a result, all the trajectories in the preprocessed data set lack such event, although they are appropriately labeled. Note that different symbolization rules, or just a permutation of Table 9.1, would yield different projected trajectories that are equally valid.

9.2.3 Task-specific data sets

As previously mentioned, we tackle two main prediction tasks: the classification of whole trajectories as they are, and the early prediction of such trajectories using only the information provided by the first T actions. We generate different sub-data sets to perform the necessary conditioning of the clickstreams for each of the tasks.

Data set (A): For the whole trajectory classification, we only keep trajectories with length $L \geq 5$ clicks. This way, we ensure that trajectories are long enough to provide statistically meaningful information in the extracted features. The resulting data set contains 203,080 trajectories: 8,324 (4.10%) belonging to class C and 194,756 (95.90%) belonging to class NC.

Data set (B): For the early classification experiment, we, again, consider only trajectories with length $L \geq 5$ clicks. Then, for each early window T to be considered, we generate a new sub-data set taking the trajectories with length $L \geq T$ and trimming them all to length $L = T$.

In both cases, we also remove excessively long to be considered meaningful or generated by humans given the limited session time. Thus, we take trajectories with length $L \leq 155$ clicks, which implies a 1% drop of Data set (A).

9.3 Hand-crafted, feature-based classification

In this section we describe the different feature-engineering schemes that are followed to extract meaningful information from the trajectories for the posterior classification. Then, we proceed with the introduction and evaluation of the different ML models used for the clickstream prediction.

9.3.1 k-grams

The most straightforward feature that can be extracted from a symbolized time series is the joint distribution of symbols. A k-gram [441] is defined as a block of k consecutive symbols. As such, isolated symbols are 1-grams, whereas blocks of two consecutive symbols such as '11', '12', '63' are examples of 2-grams.

In general, the normalized frequency histogram of k-grams is a joint probability distribution $P(s_1, s_2, \ldots, s_k)$. Asymptotically, the set of k-grams distributions $\{P(s), P(s_1, s_2), \ldots, P(s_1, s_2, \ldots, s_k)\}$ provides all the information of the dynamical process generating the trajectories. As a matter of fact, the so-called Shannon's entropy rate of such a process is defined as the limit

$$H = \lim_{k \to \infty} \frac{-1}{k} \sum_{\mathbb{B}(k)} P(s_1, \ldots, s_k) \log P(s_1, \ldots, s_k),$$

where $\mathbb{B}(k)$ enumerates all blocks of size k, i.e., all possible k-grams. The entropy rate may be used to estimate the complexity of the process underpinning the observed trajectories. However, this quantity is generally intractable, in practice, as the number of possible k-grams increases exponentially. For instance, in this case with five symbols (1, 2, 3, 4 and 6), there is a total of 5^k possible different k-grams. Therefore one would need extremely long trajectories to accurately estimate even low order approximations to H. For these reasons, given the typical session lengths in e-commerce applications, we are restricted to short k-grams, usually $k = 1, 2$.

For parsimony, the first question to address is whether the relative abundance of isolated symbols (1-grams) is already a good discriminative feature of the two

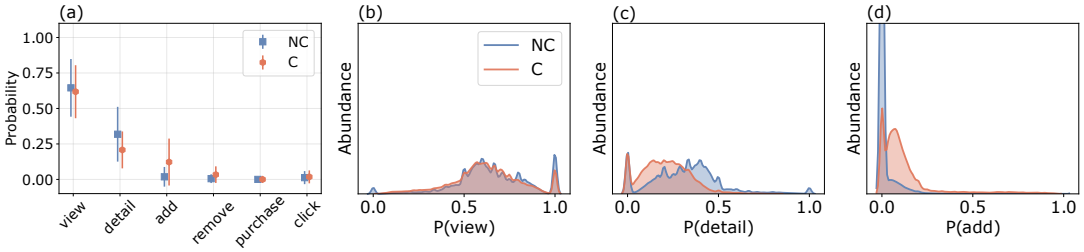

FIGURE 9.3: **1-gram distribution by class in Data set (A)**. (a) Mean normalized frequencies of 1-grams with error bars denoting the standard deviation. (b,c,d) Intraclass distribution of frequencies for 1-grams corresponding to `view`, `detail`, `add`, respectively.

classes, as in that case there is no need to look at more sophisticated patterns in the data. Intuitively, we may expect certain actions to be indicative of the class. For instance, adding products to the cart is a prerequisite for buying, although there may be trajectories without such action that belong to the C class due to the sessionization. Fig. 9.3(a) shows the average abundance of each symbol $P(s)$ in both classes. These preliminary results suggest that, while some subtle differences seem to be present for `view` and `detail` events, both classes have the same abundance within error bars; thus, 1-gram statistics are not discriminative.

We explore this aspect in further depth, computing the ensemble distributions of $P(\texttt{view})$, $P(\texttt{detail})$ and $P(\texttt{add})$ in Figure 9.3(b), (c) and (d), respectively. This finer analysis confirms that the isolated actions `view` and `detail` appear with comparable frequencies in the two classes and are overrepresented in very short trajectories. On the other hand, the `add` action is more abundant in trajectories belonging to the C class, as expected, meaning that $P(\texttt{add})$ may potentially be an informative feature.

Accordingly, we extract 1-gram and 2-gram distributions $P(s)$ and $P(s_1, s_2)$ as relevant features of the trajectories. However, it is clear from Figure 9.3 that higher order statistics are needed. Since the length of the trajectories precludes the extraction of meaningful estimations of higher order k-grams, in the next Section 9.3.2, we introduce a simple and computationally efficient combinatorial metric devised to extract higher order patterns from short samples.

9.3.2 Horizontal visibility graph motifs

A time series of N points can be transformed into a so-called horizontal visibility graph (HVG) of N nodes via the so-called visibility algorithm [437, 442]. This is a method that enables the characterisation of time series and their underlying dynamics using combinatorics and graph theory.

Definition (HVG): Let $\mathcal{S} = \{x_1, \ldots, x_N\}$, $x_i \in \mathbb{R}$ be a real-valued scalar time (or otherwise ordered) sequence of N data. Its HVG($\mathcal{S}$) is defined as an undirected graph of N nodes, where each node $i \in \{1, 2, \ldots, N\}$ is labelled in correspondence with the ordered datum x_i. Hence x_1 is related to node $i = 1$, x_2 to node $i = 2$, and so on. Then, two nodes i, j (assume $i < j$ without loss of generality) share an edge if and only if $x_k < \min(x_i, x_j)$, $\forall k : i < k < j$.

HVG implements an ordering criterion to transform a time-series into a graph representation, which is illustrated in Figure 9.4(a) (see Ref. [437] for a convexity criterion that generates 'natural' visibility graphs instead). HVGs allow us to harness the power of graph theory and network science tools [443] to describe the structure of time series and their underlying dynamics from a combinatorial perspective.

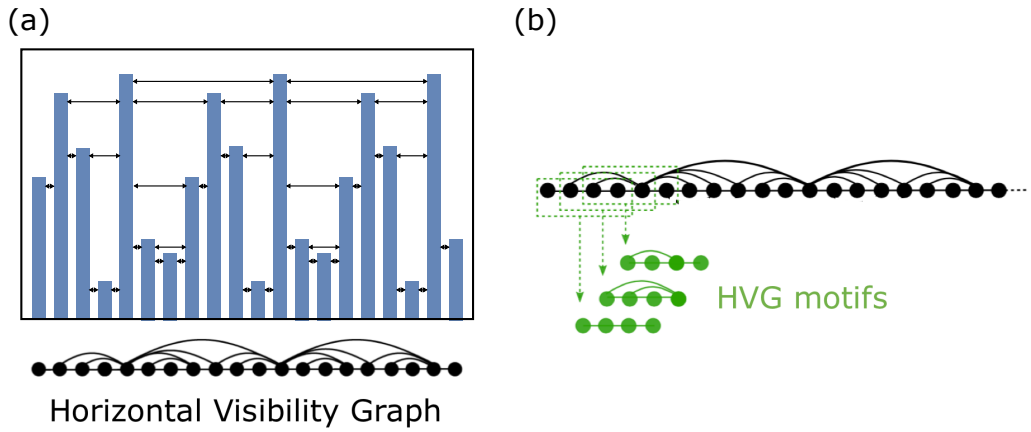

FIGURE 9.4: **Schematic representation of the generation of HVGMs** (a) Extraction of the HVG from a discrete time series according to the visibility criterion, illustrated in arrows. (b) Extraction of the sequential HVGMs of order $p = 4$ from an HVG sliding along the Hamiltonian path of the graph.

While research on this methodology has been primarily theoretical, elaborating on mathematical methods [444–447] to extract rigorous results on the properties of these graphs when associated to canonical models of complex dynamics [438, 448–450], its practical applications are diverse. In practice, HVGs serve as effective feature extraction tools for statistical learning, finding applications across multiple disciplines [451–455]. Notably, they have been recently extended from time-series analysis to image processing [456].

Since the given trajectories are relatively short, local patterns might provide more information than global metrics. This information is captured by sequential horizontal visibility graph motifs (HVGMs) [436, 457], which assess the abundance of small combinatorial structures within the time series, reflected in sub-graphs within the HVG.

Definition (HVGM of order p) Consider an HVG of N nodes, associated to a time series of N data points, and label the nodes according to the natural ordering induced by the arrow of time, i.e., the trivial Hamiltonian path. Set $p < N$ and sequentially consider all the sub-graphs formed by the sequence of nodes $\{i, i+1, \ldots, i+p-1\}$, with $i \in [0, N-p]$, and the edges from the HVG only connecting these nodes, as illustrated in Fig. 9.4(b). The resulting sub-graphs are defined as the sequential HVGMs of order p.

However, not all possible graphs with p nodes are indeed sequential HVGMs, as the lowest non-trivial order is $p = 4$ [436]. In Table 9.2 we enumerate the six possible motifs of order 4. For the specific symbolization rule defined in Table 9.1, a few examples of possible HVGMs are, for instance, `View-View-Detail-View` (1121) corresponding to the first motif, `Add-View-View-Remove` (3114) corresponding to the second motif, `Add-View-Add-Detail` (3132) corresponding to the third motif, and so on. The relation between specific motifs and behavioral patterns strongly depends on the symbolization rule, which is an issue for all categorical sequences undergoing a symbolization process.

Remarkably, HVGMs capture high-order patterns in the time series while being computationally efficient. As a matter of fact, there is no need to extract the full HVG from the time series for the posterior motif extraction, as in Fig. 9.4. Instead, HVGMs can be directly extracted in linear time checking an inequality set [436]. In Table 9.2,

Label	Shape	Inequality set
Z_1		$P(\{\forall x_0, x_1 \geq x_0, x_2 < x_1, x_3 \leq x_2\} \cup \{\forall (x_0, x_3), x_1 \geq x_0, x_2 \geq x_1\} \cup \{\forall x_0, x_1 < x_0, x_2 \leq x_1, x_3 \leq x_2\})$
Z_2		$P(\{\forall x_0, x_1 < x_0, x_2 = x_1, x_3 > x_2\})$
Z_3		$P(\{\forall x_0, x_1 < x_2 < x_0, x_3 \leq x_2\} \cup \{\forall (x_0, x_3), x_1 < x_0, x_2 \geq x_0\})$
Z_4		$P(\{\forall x_0, x_1 \geq x_0, x_2 < x_1, x_3 > x_2\} \cup \{\forall x_0, x_1 < x_0, x_2 < x_1, x_2 < x_3 \leq x_1\})$
Z_5		$P(\{\forall x_0, x_1 < x_0, x_1 < x_2 < x_0, x_3 > x_2\})$
Z_6		$P(\{\forall x_0, x_1 < x_0, x_2 < x_1, x_3 > x_1\})$

TABLE 9.2: **HVGMs of order 4.** Enumeration of all the motifs of order 4 extracted from a discrete-value time series according to a hierarchy of inequalities. The collection of all the rows provides the motif profile **Z**.

we provide the inequality sets for order $p = 4$ motifs extracted from discrete-valued sequences, see Refs. [436, 457] for additional technical details in the case of real-valued sequences. This is of utmost importance for the deployment of this type of feature extraction method with a real-time application.

We consider the motif profile **Z** as a representative feature of the trajectory. It is the simplest metric encapsulating the statistics of the HVGMs, defined as the discrete marginal distribution of the set of HVGMs. We only consider motifs of order $p = 4$, hence

$$\mathbf{Z} = [Z_1, Z_2, \ldots, Z_6], \tag{9.1}$$

where Z_i denotes the probability of the appearance of motif i. Furthermore, we compute the entropy of the motif profile to quantify its heterogeneity:

$$h_z = -\sum_{i=1}^{6} Z_i \log Z_i, \tag{9.2}$$

which increases when the different motifs are uniformly represented, and decreases whenever a particular motif is overrepresented.

9.3.3 Preface on classifier evaluation

Throughout Sections 9.3 and 9.4, we aim for a qualitative analysis of the tasks at hand, assessing the trajectory classification viability and focusing on the knowledge we can extract from the features. Hence, given that the data sets are heavily unbalanced (recall Section 9.2.3), we generate new balanced sub-data sets taking the totality of samples from the least represented class (C) and downsampling from the majority class (NC). This way, we can train and evaluate our classifiers much faster, and the class balance allows us to perform a qualitative analysis of the differences between them much more easily. We perform the benchmarking in a more realistic scenario in Section 9.5.

Then, we split the subdata set into train and test sets with a 80/20 ratio – we train the classifier on the train set and evaluate the accuracy on the test set. Furthermore, the classifier undergoes a 3-fold cross-validated (non-exhaustive) grid-search of its

most relevant parameters during the training. As evaluation metrics, we use F_1 score and area under the ROC curve, which we denote as AUC.

The whole subsampling procedure is repeated 10 times and we report the mean value of the metrics together with their standard deviation over the repetitions.

9.3.4 Clickstream prediction

We first address the full trajectory classification task using the Data set (A) (see Section 9.2.3).

Feature analysis

Before diving into the model performance evaluation, we focus on understanding the data through the features that we have engineered. We start by training an extreme gradient boosting (XGB) classifier [458] over several balanced data sets to compute the feature permutation importances, shown in Fig. 9.5(a). With this, we can obtain a better understanding of the underlying behavioural patterns that lead towards purchasing an item.

Given that symbol 3 corresponds to adding a product to the cart, we would expect its sole presence $P(3)$ to be quite significant, as discussed in Fig. 9.3. However, it is the combination of two actions `add-view` ($P(3,1)$) that is more relevant for the classification. Furthermore, this not even the most relevant feature, showing that web exploration patterns, related with actions `view` and `detail`, actually provide the most important information ($P(1,2)$). This is backed up by the importance of some HVGMs, which show that some of these navigation patterns are, indeed, quite revealing. This is extremely important, provided that trajectories are mostly composed by these two actions, as shown in Fig. 9.3, meaning that most users jump from page to page (series of `view` (1)), product to product (series of `detail` (2)) and page to product (`view-detail` (12) or reverse) but rarely dive into a specific product straight from a search (`click` (6)).

We dive deeper into the analysis with SHAP's tree interpreter [459, 460] to see the effect of each feature in the prediction. In Fig. 9.6, we see the influence of the most relevant features according to the SHAP values (see Refs. [459, 460] for details), which are in line with the previous feature importances Fig. 9.5(a). The 2-gram $P(1,2)$ corresponding to `view-detail` is the most relevant feature with high values indicating that the customer is, most likely, not buying before leaving. Combined with what we observe for the other features, behaviours such as wandering from page to page (high `view-view` $P(1,1)$), opening many products (high `view-detail` $P(1,2)$ and `detail` $P(2)$) without going straight from one to another (low `detail-detail` $P(2,2)$) and casually doing some searches (high `click` $P(6)$) are indicative that the customer at hand will end the session without buying. High values of Z_1 and low values of the h_z indicate that such trajectories are rather monotonic, which is not surprising when considering about a casual browser looking at what is offered. An example of a trajectory could be `view-view-view-detail-view-detail` (111212) providing high `view-view` ($P(1,1)$), `view-detail` ($P(1,2)$) and `detail` ($P(2)$), includes two Z_1 and a Z_4, resulting in a moderate h_z. This example would correspond to a customer that scrolls through different pages until it finds a product to look at, goes back and then checks another product.

On the other hand, behaviours such as visiting few products and many different pages (high `view` $P(1)$) but coming from different actions (low `view-view` $P(1,1)$ and high `detail-view` $P(2,1)$, `add-view` $P(3,1)$) are indicative that the user

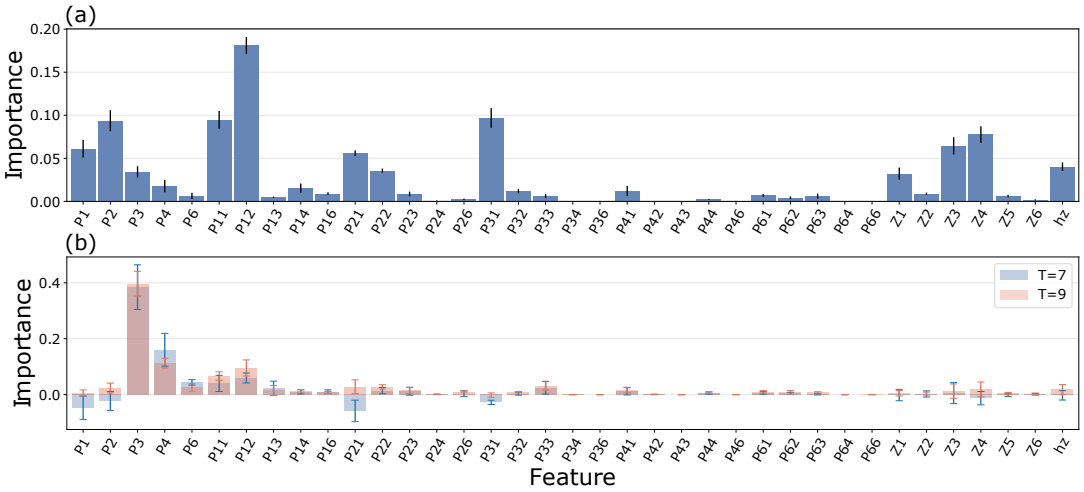

FIGURE 9.5: **Feature importances of an XGB classifier.** From left to right there are the permutation feature importances of the 1-grams, the 2-grams, the HVGMs and their entropy h_z for (a) Data set (A), and (b) Data set (B) for early windows $T = 7, 9$. For readability we have omitted the parentheses in $P(s)$, $P(s, s')$.

will purchase an item before leaving. Other recognisable patterns involve going straight from one product to another (high `detail-detail` $P(2,2)$), which suggests that these users may take advantage of features such as similar product recommendations. High values of the HVGM entropy h_z also indicate that navigation patterns are richer. An example of a trajectory with such properties could be `view-add-detail-view-add-view` (132131) with low `detail` $P(2)$ and `view-view` $P(1,1)$, but with high `view` $P(1)$ and `add` $P(3)$, as well as `add-view` $P(3,1)$ and `detail-view` $P(2,1)$. It contains a Z_1, Z_3 and Z_5, maximizing the pattern entropy h_z. Overall, this is indicative that customers that purchase an item tend to have a prior idea of what they want.

It remains to be explored whether different symbolizations from the one in Table 9.1 may highlight different behavioural patterns in terms of the HVGMs. Z_1 is the motif that contains the most possible patterns and thus it is the most common. Other HVGMs are more restrictive and represent more specific scenarios, such as Z_3 (indicative of class C) and Z_4 (indicative of class NC). With other symbolizations, patterns that are currently compressed in Z_1 may fall into other motifs, which might turn out to be characteristic of one class or another.

Pipeline definition and evaluation

We consider five classifiers: (i) a logistic regression (LR), (ii) a RF [58], (iii) a support vector machine (SVM) [461], (iv) an XGB [458], and (v) a shallow dense neural network (NN). Each pipeline originally contained an additional pre-processing step, where we explored using two different types of dimensionality reduction: principal component analysis (PCA) [462] and uniform manifold approximation and projection (UMAP) [463], subsequently followed by the classifier. However, we have found it is best to refrain from any dimensionality reduction for this task, as it hurts the performance on the validation sets. This removes a significant computational workload, especially in the case of UMAP, making the whole process much faster. Between the feature extraction and the classifier, the only pre-processing that is left is a normalization step subtracting the mean and dividing by the standard deviation for the NN-based classifier.

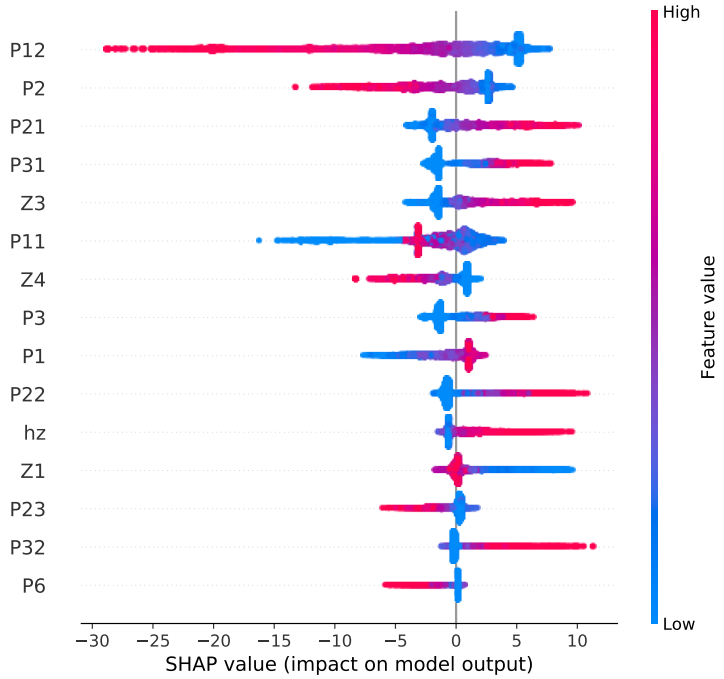

FIGURE 9.6: **Influence of the top fifteen features and their SHAP values with an XGB classifier.** Each dot in the plot corresponds to a sample. On the y-axis, features are listed in order of relevance, from top to bottom. The x-axis indicates the influence of the feature in the classification of the given sample. The point color indicates the relative value of the feature for each given sample. Hence, the plot shows how the different values of every feature contribute to the prediction of every sample: left is NC and right is C.

We report the classification performance metrics in Table 9.3. The best classifiers output F_1 scores around $87 - 88\%$. The NN holds a significant advantage in terms of AUC with respect to the rest, suggesting that it has a much better notion of separability between the two classes and that most errors come from overlapping samples of different classes. In principle, since the HVGM profile depends on the symbolization rule, higher scores could potentially be achieved by finding a rule that enhances the discrimination between classes. However, we lack a direct way of finding such optimal configuration a priori, so we simply consider the obtained classification scores as a lower bound, and we leave the general task of finding the optimal symbolization rule as an open problem for future research.

Classifier	F_1	**AUC**
LR	$84.05 \pm 0.61\%$	$84.67 \pm 0.52\%$
RF	$87.65 \pm 0.84\%$	$87.70 \pm 0.82\%$
SVM	$87.46 \pm 0.40\%$	$87.65 \pm 0.38\%$
XGB	$88.08 \pm 0.39\%$	$88.17 \pm 0.35\%$
NN	$\mathbf{88.17 \pm 0.61}\%$	$\mathbf{94.53 \pm 0.28}\%$

TABLE 9.3: **Full trajectory classification results with feature engineering.** Mean F_1 score and AUC over 10 subsamplings of Data set (A) $\pm$ standard deviation for the different classifiers.

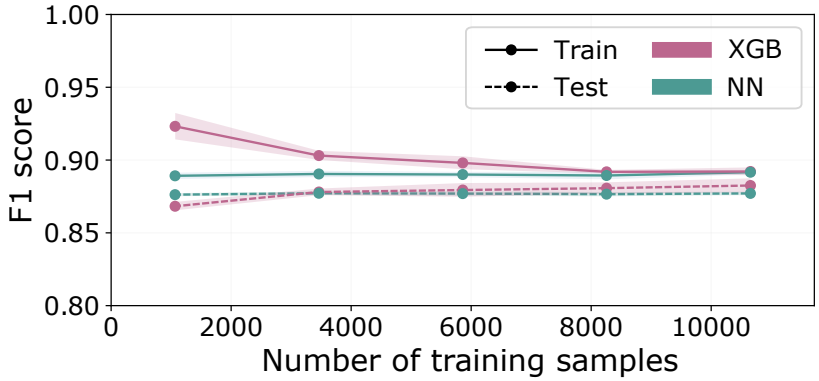

FIGURE 9.7: **Learning curves of the best perfomring classifiers.** The models are evaluated through nested cross-validation of 5 outer and 3 inner folds to optimize their hyperparameters. The curves show the mean accuracy over the 5 folds and the shaded area is the standard deviation for both the train and test sets as function of the amount of data used.

Learning curves

In an industrial application, the database is continuously growing as new data is acquired with customers browsing the online shop. Therefore, it is critical to assess the performance and correct behaviour of the models under such conditions, in order to know what can be expected in a near future and, more importantly, whether there is room for improvement.

In Fig. 9.7, we show the learning curves of two of the best performing classifiers: the NN and the XGB. The NN performance is barely unaltered with additional data, displaying a nearly constant gap between train and test scores. In contrast, the XGB classifier improves continuously with further amounts of data. However, it does so with diminishing returns, rendering the impact of additional data beyond $\sim$ 6000 samples insignificant. Furthermore, it comes at the expense of longer training times for this model.

Both models are stable and benefit from additional training data, even if only marginally. This means that both models already achieve competitive performances with small amounts of data, which enables the application of such systems even in the early stages of e-commerce websites. As a final remark, none of the classifiers is overfitting despite not using any kind of dimensionality reduction.

9.3.5 Early prediction

In real-time retail applications, early predictability of the user intent plays an essential role in targeted marketing schemes and other strategies. This involves performing a trajectory classification task with the limited information gathered as the user interacts with the website. Given that the feature extraction is scalable and efficient, finding evidence of early detection opens up the possibility of implementing the pipeline in real time.

To tackle the early prediction task, we work with Data set (B), in which trajectories cut after the initial T points simulate a user that has only triggered T events since the beginning of the session. We study the effect of the “earliness" in the detection by varying the early window T in the range $T = 5, \ldots, 14$. We quantify “how early" in the trajectories the prediction is performed relative to their lengths with an

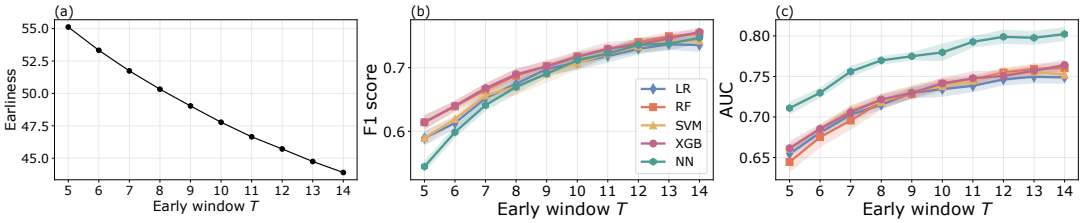

FIGURE 9.8: **Early trajectory classification results.** (a) Earliness parameter for the data sets generated for each early window T. (b,c) F_1 and AUC (respectively) for each classifier as a function of the early window T, where only the first T data points of each trajectory are used to extract the features.

earliness parameter defined as:

$$\text{Earliness} = 100\left(1 - \frac{T}{\Omega}\sum_{i=1}^{\Omega}\frac{1}{L_i}\right), \tag{9.3}$$

where L_i is the size of the i-th trajectory and Ω is the total number of trajectories. As illustrated in Fig. 9.8(a), when the early observation window T is large, Earliness $\to 0\%$. As the window shrinks, the earliness parameter increases (the limit value depends on the actual size of all trajectories L_i). More precisely, it provides the average relative distance between the point at which the prediction is performed with respect to the length of the trajectories.

While we have seen that navigation patterns are relevant for the classification of full trajectories, these do not have enough time to emerge clearly in the intrinsically short trajectories of the early classification. Hence, we expect different features to become more relevant in their place. Similar to Section 9.3.4, we conduct a brief feature analysis computing the permutation feature importance for different early windows $T = 7, 9$, shown in Fig. 9.5(b). For very short time windows, the relevance of the features varies significantly between repetitions and the XGB classifier is confused by some of them such as $P(1)$ and $P(2,1)$. As the observation window grows, the statistical features can be better estimated and the model can make better use of them, reflected in the stabilization of the feature importance for $T = 9$. For such short early windows, we observe that the most relevant features are the presence of `Add` ($P(3)$) and `Remove` ($P(4)$) actions. Intuitively, managing items in the cart early in the session is very revealing of the user's intentions.

We show the early classification results in Fig. 9.8(a) and (b), where we find that early prediction of customer intent is indeed possible. The best models reach F_1 scores beyond 60% for $T = 5$, and go beyond 70% with only $T = 9$ points. Interestingly, the NN classifier reaches significantly higher AUC than the rest, although it yields the lowest F_1 for the shortest windows. On average, the predictions are conducted with around 50% anticipation, as we see in Fig. 9.8(a), meaning that the actual live sessions are, on average, more than twice as long. This provides the system with enough time to react to the user's behaviour.

9.4 From hand-crafted to automatic feature extraction with deep learning

To complement the previous analysis, we also consider two DL-based approaches to the clickstream prediction challenge. In this section, we introduce the different

methods considered here and evaluate them for both tasks: full trajectory classification and early classification.

9.4.1 Deep learning models

As is often recognized in the literature [427, 428], there are strong resemblances between the clickstream prediction problem and standard tasks in the field of natural language processing (NLP). Mainly, in both cases we consider sequential data comprised of discrete abstract concepts. Hence, we draw inspiration from typical solutions derived in the field of NLP to build our models. As a baseline, we implement a Markov chain classifier, a popular choice for NLP tasks before the DL revolution. Then, we train two RNNs comprised of LSTM layers [63], introduced in Section 2.2.2: a generative model that learns the probability distribution of the data, and a purely discriminative one.

Markov chain classifier

To build our baseline, the trajectories are first separated by class, as an independent k-gram Markov chain needs to be trained for every class. Then, we extract k-grams from the trajectories to build a transition by counting how often the last event in a k-gram co-occurs with the preceding ones in the same k-gram. Normalizing the co-occurrence frequency counts, we obtain the conditional probabilities of each event with respect to the previous k points. After a hyper-parameter search, we use $k = 5$ (see details in Ref. [423]).

The trajectory classification is performed using the Bayes rule. For instance, consider the conversion class C. The probability of finding this class $p(\mathrm{C})$ in the training corpus is the *prior*, whereas the conditional probability of observing a given trajectory $\mathcal{S}$ under the Markov chain trained on C sequences is the *likelihood* $p(\mathcal{S}|\mathrm{C})$. Analogously, the conditional probability that the trajectory belongs to the NC class is $p(\mathcal{S}|\mathrm{NC})$. Hence, the *posterior* reads

$$p(\mathrm{C}|\mathcal{S}) = \frac{p(\mathcal{S}|\mathrm{C})p(\mathrm{C})}{p(\mathcal{S}|\mathrm{C})p(\mathrm{C}) + p(\mathcal{S}|\mathrm{NC})p(\mathrm{NC})} \,. \tag{9.4}$$

In this way, the classification is performed by evaluating Eq. (9.4) and assigning the class C whenever the posterior is above a certain threshold value. Here, we set the threshold to be 0.5, without performing any optimization.

Essentially, we derive two generative models: one for C and one for NC clickstreams. The classification is performed by evaluating the trajectories with both models and selecting the class corresponding to the one that assigns the highest probability weighted by the priors. This is also referred as the one which is the *least surprised*. Finally, since the prior probability for each class changes as a function of the sequence length, we compute length-sensitive priors dividing the number of C trajectories of a certain length by the total number of trajectories of the same length. In order to prevent sparsity, we bin lengths by one up until $L \leq 50$, by ten for lengths $50 < L \leq 100$ and by twenty-five for lengths $100 < L \leq 150$. Lengths longer than $L \geq 150$ fall into the same bin.

Generative LSTM classifier

The state-of-the-art method at the time of conducting the research builds upon the previous success of the Markov chain approach [427], enhancing it with RNNs [428].

Classifier	F_1	AUC
Markov ($k = 5$)	$89.45 \pm 0.61\%$	$95.51 \pm 0.45\%$
genLSTM	$90.21 \pm 0.36\%$	$96.01 \pm 0.36\%$
discLSTM	$\mathbf{91.03 \pm 0.48}\%$	$\mathbf{96.83 \pm 0.37}\%$

TABLE 9.4: **Clickstream classification results for the DL models.** Mean F_1 score and AUC $\pm$ a standard deviation over 10 subsamplings of the data set for the different classifiers.

Keeping the same principle, the Markov chains are replaced by LSTMs that provide the probability distribution over the next possible actions given the previous elements of the clickstream at every time step. These models are trained in a supervised way by predicting the next event at every time step of the training trajectories, just like a typical language model. Hence, we train one LSTM for each class, as in the previous case, although we consider two additional events besides the actions summarized in Table 9.1: a beginning and end of sequence event.

Again, the classification is Bayesian: the trajectories are evaluated by both models, retrieving the the probabilities of every event along the way, which are used to evaluate Eq. (9.4). The length-sensitive priors are computed as described in the previous section.

Discriminative LSTM classifier

We propose a different way to tackle the clickstream problem, inspired by the generative LSTM method. Rather than training a generative model for every class for the subsequent Bayesian classification, we directly train a discriminative model from the beginning. The model uses an LSTM layer to extract meaningful features from the trajectories by parsing them sequentially. Then, these features based on the LSTM's hidden activation are directly fed into a dense feedforward classifier that outputs the probability that the trajectories belong to either of the classes. This model is trained as any regular classifier, without attempting to learn the probability distribution of the data. This approach follows the same principles as the methods used to characterize anomalous diffusion [364, 369] presented in Chapters 6 and 7.

In Ref. [423], we explore two methods to extract the trajectory features with the LSTM: averaging the hidden activations along the trajectory, and taking the last hidden state at the end of the trajectory. Both provide similar results, with the latter being slightly better overall. Therefore, we only show the results for this one in this thesis.

9.4.2 Clickstream prediction

As in Section 9.3.4, we start by addressing the clickstream prediction problem with full trajectories.

We train the three methods in Data set (A) (recall Section 9.2.3) and evaluate their performance in terms of the F_1 score and AUC, which are reported in Table 9.4. Overall, there is a significant performance improvement with respect to the feature engineering results reported in Table 9.3, with the Markov chain baseline already outperforming the best feature-engineering pipelines. The generative LSTM model (genLSTM in the table) does, indeed, improve over the Markov chain, although the biggest improvement is seen with the proposed discriminative LSTM (discLSTM in the table), establishing a new stae of the art for the task.

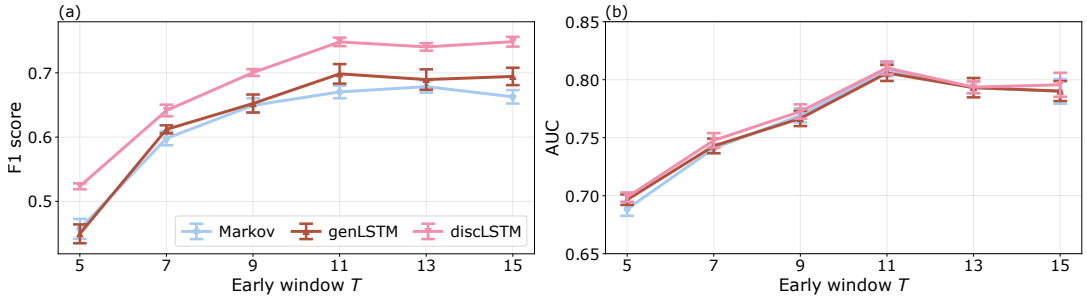

FIGURE 9.9: **Early clickstream prediction with DL methods** (a) F_1 score and (b) AUC as function of the early window size for early prediction.

9.4.3 Early prediction

Moving on to the early classification task, as in Section 9.3.5, we train and evaluate the models using the data in Data set (B). Again, we consider multiple early detection windows that mostly fall within the first half of the trajectories.

Fig. 9.9(a) and (b) show the F_1 score and AUC, respectively, for the different methods. The results are comparable to the ones obtained with handcrafted features shown in Fig. 9.8 and, in fact, the feature-engineering schemes outperform all the models introduced in this section in terms of F_1 score. Only the discriminative LSTMs achieve similar performance for long windows. Nevertheless, the DL-based models hold the advantage in terms of the AUC, although the shallow NN feeding on the features achieves the same AUC or arguably better as we discuss below.

As in the full trajectory classification, the generative LSTMs are outperformed by the discriminative ones. All models have similar AUC with meaningful differences only emerging for $T > 13$, as depicted in Fig. 9.9(b). Interestingly, the curves are non-monotonic and peak at $T = 11$, exhibiting a slightly decreasing trend thereafter. It seems counter-intuitive that providing the models with more information results in lower performance. One possible explanation may reside in the size of the training data, which is reduced as the early window increases, since only trajectories of length $L \geq T$ are considered in each sub-data set. The observed behaviour may be explained by the combination of two factors: increasing the trajectory information and decreasing the amount of training samples. This is not observed in the feature-engineering models, which exhibit a clear monotonic tendency (Fig. 9.8(c)) and are robust to the data set size (Fig. 9.7), which is why the AUC obtained by the NN classifier is arguably better than the ones achieved here.

9.5 Benchmarking on realistic scenarios

In this section, we evaluate the different proposed methods with a more realistic approach. We consider several sub-data sets with different class balances that range from the 50:50 (C:NC proportion) balance shown in the previous sections, to the original data set balance around 4:96. With this approach, we cover a wide spectrum of cases, provided that class balance may be a major determinant of model performance and it is subject to great variability [440]. Unlike in previous cases from Sections 9.3 and 9.4, the models are trained with whole trajectories and are used for both the full clickstream and early predictions. This procedure aims to replicate the requirement that would be involved in a real business setting, in which a single model is asked to make predictions at different time steps.

Classifier	F_1	AUC
LR	$40.41 \pm 0.50\%$	$77.43 \pm 1.29\%$
RF	$45.77 \pm 0.63\%$	$73.07 \pm 1.84\%$
XGB	$49.87 \pm 0.58\%$	$79.87 \pm 0.84\%$
NN	$\mathbf{50.78 \pm 0.27}\%$	$\mathbf{94.75 \pm 0.07}\%$

TABLE 9.5: **Handcrafted-feature-based classifier performance when trained with 4:96 class balance.** Mean F_1 score and AUC over 10 train/test partitions of Data set (A) $\pm$ standard deviation. The models are trained and evaluated with the original class balance of 4:96 following a class re-weighting strategy instead of downsampling the majoritary class. We have refrained from training the SVM as it is extremely costly for such amount of samples.

To generate the different scenarios, we down-sample the majority class to obtain the desired class balance in the validation and test sets, but train in balanced data sets. In Ref. [423] we explore different strategies to perform the training and we find that downsampling to fully balanced datasets provides the best trade-off between accuracy and training time. However, the focus here is on the impact of the class imbalance during the testing, regardless of the training procedure as long as it is consistent. We perform ten repetitions of the whole process in order to account for the stochasticity in the down-sampling processes and the models are trained and evaluated without any hyper-parameter fine-tuning.

9.5.1 Clickstream prediction

We first analyze the different model performances in the classification of full trajectories. Fig. 9.10(a) and (b) show the F_1 score and AUC, respectively, for all the classifiers.

Notably, the discriminative LSTM model provides the best overall performance in terms of both the F_1 and AUC. On the other end, the LR classifier consistently provides the worst metrics, which suggest that it does not have enough capacity to fully capture the complexity of the task. The feature-based NN classifier suffers the most from class imbalance in terms of F_1, and the AUC becomes rather unstable for highly imabalanced data sets. Nevertheless, the sheer discrepancy between F_1 and AUC suggests that it is most likely an issue related to the classification threshold not being optimized for the test imbalance, given that high AUC values are indicative of good class separability.

In light of the results, we train the feature-based classifiers with the original class balance without down-sampling the training set and, instead, we opt for a class re-weighting strategy. With this, we significantly improve the performance in the 4:96 data set by more than 10% F_1 for the XGB and NN classifiers, as shown in Table 9.5. With this training scheme, the resulting models (barring the LR classifier) outperform the generative models. The fact that all models keep a relatively high AUC throughout the different class imbalances shows that all methods are capable of correctly separating both classes. Nonetheless, increasing the amount of samples of the majority class (NC) increases the classification noise at the class boundary, as well as the amount of duplicate samples in feature space belonging to different classes. The NN classifier trained with the original class balance reaches a $\sim$ 90% true positive rate at about $\sim$ 10% false positive rate with an AUC of 94.71%. Nonetheless, a 10% of the majority class is already twice as large as the whole minority class corpus.

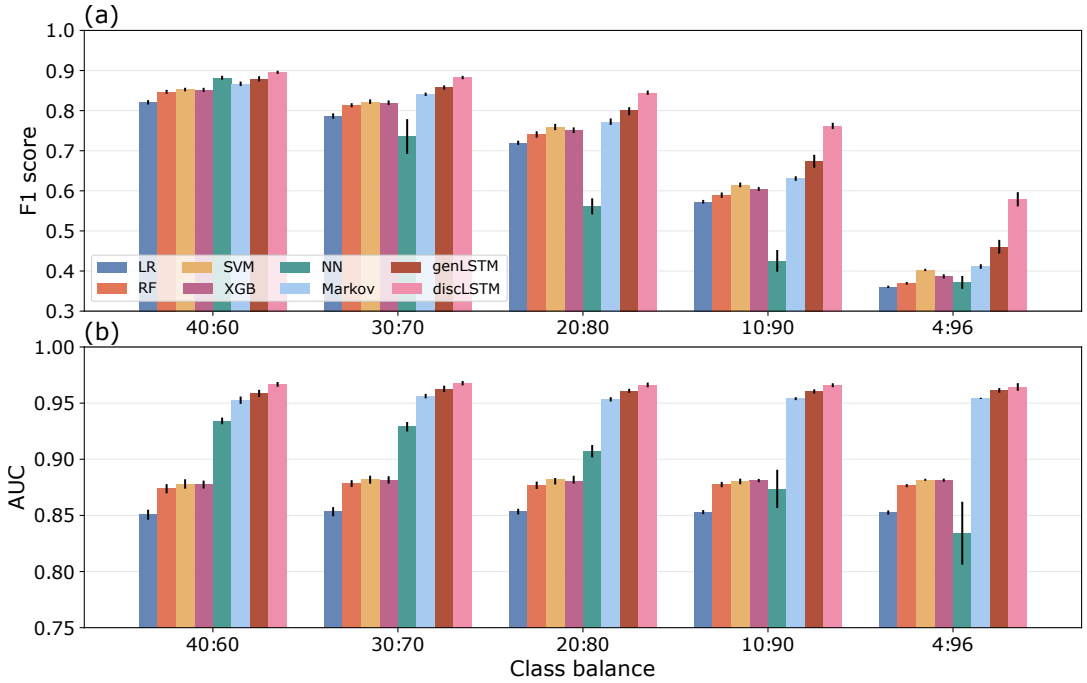

FIGURE 9.10: **Benchmark on full clickstream prediction at different class balances.** (a) F_1 and (b) AUC results across multiple scenarios featuring different class balances C:NC that are closer to real applications. All models are trained on balanced data sets.

Overall, the discriminative LSTM outperform the rest. For mild class imbalance, the performance is comparable among all methods, and the differences become more evident as the class imbalance increases. Among the feature engineering models, the NN and XGB are the best performing classifiers, which we have shown that outperform the generative methods when properly trained even in high class imbalance cases. This shows that the HVGM features improve a simple 2-gram feature representation to the point that it outperforms a statistical language model that leverages transition probabilities in k-grams up to $k = 5$. Among the DL models, the discriminative method outperform the rest.

9.5.2 Early prediction

Finally, we address the challenge of early clickstream prediction. In order to simplify the analysis and prevent cluttering the exposition, we drop the SVM and the RF classifiers. The RF classifier is consistently outperformed by the XGB one and both are tree-based methods. The SVM is consistently outperformed by the XGB and NN classifiers. Hence, we keep the simplest and the two best classifiers, to provide a sense of the range of performance.

Several different clickstreams can share the same initial sequence of events, while belonging to different classes. For this reason, we devise an Oracle model that knows the true distribution of classes assigning the empirical probability that a sequence belongs to either class to every trajectory in Data set (B). For example, consider a trajectory that is repeated a total of five times, four belong to class C and one to class NC, the model will assign it the empirical probability of 0.8 (4/5) to belong to the C class. For early prediction, the shorter the early window, the higher the probability of having duplicates, which constitutes one of the major challenges of the task. This model provides a reference to assess whether the models are performing close to optimally. Nevertheless, while this model maximizes the accuracy, it does

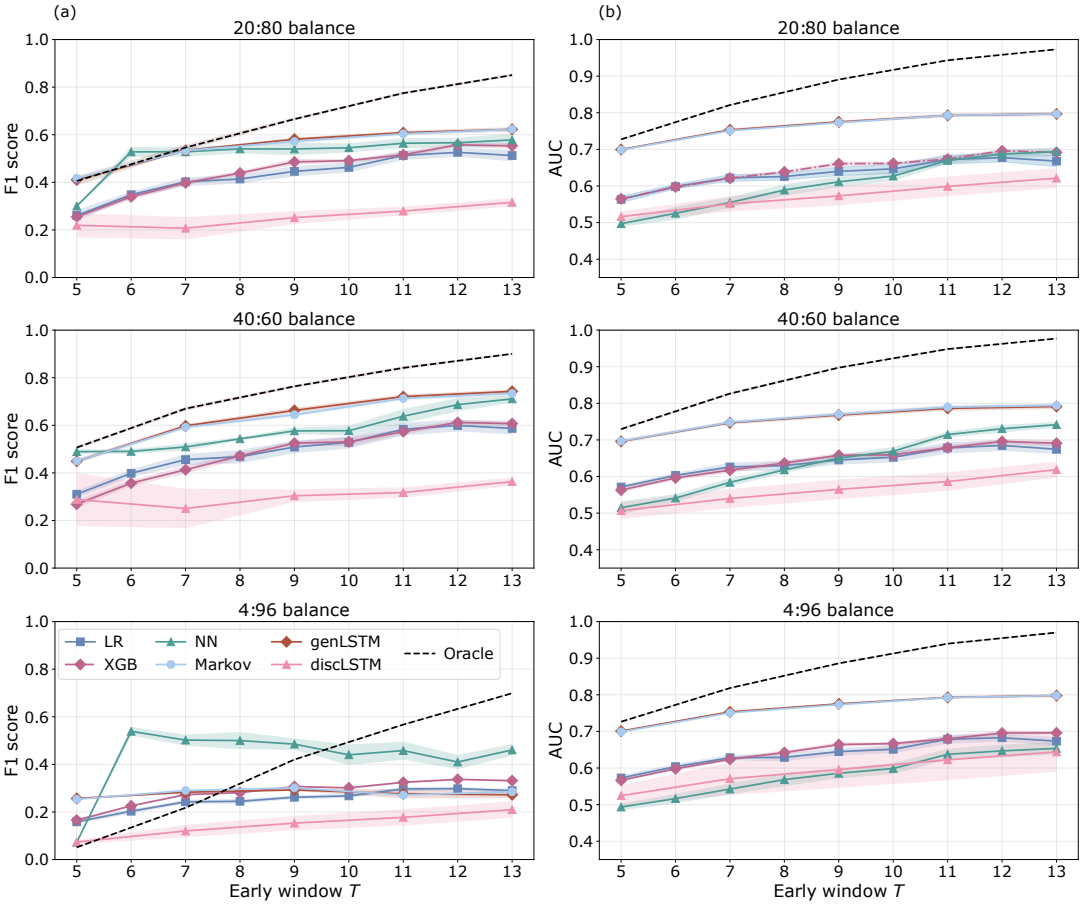

FIGURE 9.11: **Benchmark on early clickstream prediction at different class balances.** (a) F_1 and (b) AUC results on early prediction task in various scenarios with different class balances (C:NC). All models are trained to perform full trajectory classification on balanced data sets.

not necessarily maximize the F_1 or AUC, as the degree of uncertainty enters into play and class imbalance heavily affects results. We have refrained from plotting the Oracle prediction in Fig. 9.10 provided that its AUC is close to one and that F_1 ranges from 96% to 93% with increasing imbalance, meaning that full trajectories are rarely repeated.

Fig. 9.11 shows the results on early prediction for various class balances. Perhaps the most noteworthy result is that the discriminative LSTM model that topped the performance in all the previous studies is, by far, the worst performing method. This is the model that suffers the most when facing a different task from the one for which it is optimized (full clickstream classification). The generative LSTM and Markov chain consistently provide the best results until the highest class imbalance is reached, where the feature-based approach takes the lead.

These two classifiers seem to suffer less from the discrepancy between the training objective and the test evaluation, as they do not attempt to learn the relationship between sequences of events and classes directly, but, rather, they develop a generative model of how users behave when they aim to purchase and when they do not. However, these models have their own limitations as well, the main one being that a separate model needs to be trained for every different class, with MAP classification on top of it. Therefore, they are appealing when dealing with few classes (as in this case), but do not scale well to tasks which involve too many of them.

These results highlight a theoretically and practically relevant difference between generative and discriminative models. The former models are slightly less efficient but learn to extract more robust representations of the trajectories and, thus, generalize much better to other tasks. Conversely, the latter models are highly specialized and provide the best results when evaluated on their target task, at the cost of poor generalization, which is reasonable. Similar observations have motivated the biggest breakthroughs in NLP, which come from the combination of training generative models to extract meaningful information that are then leveraged to perform downstream tasks [464]. The poor generalization of the discriminative model may indicate that the most discriminative sequential patterns appearing towards the end of the clickstreams. The model learns to capture them with full trajectories, achieving good results but neglecting any other possible relevant information. When faced with only the initial points of the trajectories, these patterns to not appear and, thus, the model fails. Nevertheless, the positive results from Fig. 9.9 prove the existence of discriminative patterns early on, although they are likely to differ from and be less predictive than the patterns learned on whole sequences.

Overall, the feature-engineered models are more resilient to class proportions providing the best results in the most realistic scenario with the strongest class imabalance. Noticeably, the NN classifier provides the best F_1 score (Fig. 9.11(a)) with some of the lowest AUCs (Fig. 9.11(b)). This resilient behaviour makes feature-engineerd models appealing for industrial applications. Besides, they are a powerful tool to interpret potentially useful patterns in clickstream sequence that can help to understand the performance of more efficient and opaque models, such as the DL-based ones.

Other directions that could be explored would be training the models to predict at a given early window size and see how they generalize to other inference times, instead of training them with full trajectories. Furthermore, future research should investigate whether the strengths of discriminative and generative models can be combined, for example, using multi-task models which jointly optimize parameters to predict the next event and the sequence class.

9.6 Conclusion

In Ref. [423], we leverage a new clickstream data set from a popular e-commerce website to address the clickstream prediction challenge. Our results show that it is possible to reliably predict conversion based on the most basic information (e.g. event type) using simple and lightweight features in a variety of scenarios. It is even possible to do so with the data of just a few user interactions with the website. On one hand, we provide algorithms that improve upon the state-of-the-art (at the time of conducting the research) ML systems for the clickstream prediction challenge. On the other, we draw a general map of this sequence-classification problem, by providing insights, benchmarks and solid baselines both at the computational and implementation levels.

We compare different approaches to sequence modelling, and tackle the clickstream challenge as a full sequence as well as an early prediction scenario. We first introduce ML pipelines based on feature engineering. Through extensive benchmarks and model interpretation techniques, we gain insights about behavioural patterns of e-commerce customers. Finally, we provide DL-based methods that provide consistently positive results for trajectory classification, at the cost of increasing training time and more complex engineering infrastructure. Since no size fits all, and

different players have different constraints, our quantitative comparison highlights precisely the trade-off between the two algorithmic approaches, so that industry decisions can be made with known bounds on performance. Our extensive analysis of generative and discriminative models in the minimal, symbolized setting produced not only an "algorithmic recipe" to solve this particular challenge, but also more general insights on the underlying informational structure, and, possibly, methodological considerations to be ported to other challenges involving symbolized sequences.

Three directions for future work can be outlined. First, improving performance by including more information: by using more sophisticated features and/or by extracting more metadata to build the symbolic sequences. For example, the prediction accuracy is likely to be improved by coupling k-grams and other network structural properties associated to the location of the trajectory at each time step. While preliminary experiments with time intervals between events only yielded marginal accuracy improvements, combinations of product metadata and time intervals are worthy of further investigation.

Second, extend to other classification problems: predicting purchase is only one of several user intent prediction problems of relevance for the industry. For instance, other possibilities include cart abandonment prediction, that is, predicting whether a user that has loaded the cart with some products will, ultimately, purchase some of them or leave the session without purchasing anything. This problem is of utmost relevance and can be readily tackled under our framework by focusing on all the trajectories where the `Add` action, corresponding to adding a product to the cart, appears, at least, once without a `Purchase` action.

Third, implementing efficient prediction models, such as the ones described in this paper, for online testing. By doing so, the algorithm could proactively detect customers which have a high probability of leaving the session without purchasing and, then, for instance, implement targeted marketing strategies to boost their conversion likelihood.

Chapter 10

Conclusion

This thesis has been a journey through different scientific disciplines and applications with machine learning (ML) as a conducting theme. After a thorough introduction of its fundamentals, we have embarked on an exploration of prominent challenges in quantum physics, both in fundamental studies and in the development of cutting edge quantum technologies. Then, we have ventured in the field of diffusion, where we have tackled some of the main issues in the study of diffusion processes in both biophysical and industrial applications.

In the first part of the thesis, we have shown how to leverage reinforcement learning (RL) in the study of quantum many-body physics using examples of paradigmatic tasks in condensed matter physics and quantum information processing. The framework we propose in Chapter 4 not only allows the systematic relaxation of complex problems, but also enables their autonomous exploration with transfer learning. Furthermore, our general formulation of the constraint space allows the application of other optimization methods besides RL, which can yield faster results in some cases. As future work, it remains an open question studying the tight relationship between the optimal relaxation properties and the physical properties of the system. Additionally, the optimization framework can be most valuable in cases where there is more than one possible direction to tighten the relaxations, such as with inflation methods.

Quantum computers bring many promises together with exciting challenges at multiple levels. In Chapter 5, we showcase another application of RL to quantum technologies. However, this time, we tackle an engineering task, where the main goal is to ensure that superconducting quantum computers perform at the best of their hardware capabilities. To this end, we implement an RL scheme to design custom controls for every qubit in the device, which results in highly-accurate and robust gate executions. Such advances enable the use of current noisy intermediate-scale quantum (NISQ) computers at larger scales to conduct better research on them. As future work, the proposed framework should be brought from simulations to real devices to better understand its practical strengths and limitations.

Most of the problems that we have considered in our exploration of the quantum many-body physics field involve the exploration of vast hypothesis spaces, such as the constraint space, or all the possible combinations of the quantum computer controls. These are akin, or even dwarfed, by the spaces of all the possible board combinations in Go, or the 3-dimensional tensor decompositions in unit rank tensors, which have been dealt with in outstanding RL applications. The capabilities of RL to explore and make sense of these vast hypothesis spaces has allowed us to develop novel methods to consistently overcome common challenges in quantum physics.

In the second part of the thesis, we have shown how to use ML techniques typical from natural language processing (NLP) and signal processing to characterize

diffusion processes and the emergence of anomalous diffusion. In Chapter 7, we introduce a method to study anomalous diffusion with constant properties. By incorporating further advances in the field of ML, we have evolved the method to process trajectories with time-dependent diffusion properties in Chapter 8. This allows us to extract richer information from the data, allowing us to extract biophysical parameters from experimental data that were previously inaccessible. Nevertheless, there is much room for improvement in the field. For instance, the single-trajectory characterization paradigm can be enhanced by considering the experiments with heterogeneous trajectories as a whole, using all the trajectories as context to individually characterize single trajectories, which we leave as future work.

Finally, the study of diffusing particles is not limited to actual physical entities. In Chapter 9, we study the diffusion of internet users through their action space for an industrial application. There, we shift our focus to engineering meaningful features in order to obtain a deeper understanding of the behavioural patterns of internet users. We show that it can be determined whether a user will make a purchase before closing the session with very few interactions between the user and the website, and with minimal action information. As future work, it remains to be seen what additional information can be added that provides the maximum amount of information, while keeping it at the minimum.

The various ML techniques employed in the analysis of sequential data related to diffusion have proven essential to successfully conduct the desired tasks. It is clear that they play a pivotal role in the understanding of diffusion processes, and they will become even more prominent as more practitioners in the field adopt and improve them.

Overall, ML and, more generally, artificial intelligence (AI), have demonstrated to be fundamental research tools that allow us to push the frontiers of what is deemed possible. The further development of AI, in combination with the adoption of domain specific knowledge, holds the potential to elevate scientific discovery to new heights. However, while AI plays such an important role, many researchers, experts in their respective fields, lack the knowledge to understand it and use it. Collective initiatives such as the AnDi challenge, that promote large community efforts to address remarkable challenges in specific fields, allow researches to be exposed to new techniques from different fields, bridging the knowledge gap. Nevertheless, we must not forsake the development of traditional, analytical and domain-specific techniques.

Appendix A

STEP architecture and data set details

In Chapter 8, we propose STEP, a novel machine learning (ML) approach to study diffusion processes with time-dependent properties. Here, we provide technical details about STEP's architecture and the data sets used to train and evaluate the models.

A.1 Architecture details

We propose to use a model that takes a trajectory $\mathbf{x}$ as input and outputs the target diffusion properties at each time step. The input trajectory is a d-dimensional vector of arbitrary length T, whose elements, x_t, correspond to the particle position at every time step t. Then, the output is a one-dimensional vector of length T, whose elements correspond to the diffusion property of interest at every time step, e.g., D_t in the case of the diffusion coefficient. See Fig. A.1 for further details about the dimensions. Throughout this work, we mainly consider trajectories of dimension $d = 2$.

The model we propose consists of three main modules: an initial convolutional part that processes the input trajectory; a self-attention-based part that feeds on the features extracted by the previous one; a shallow pointwise fully connected feedforward module that provides the desired output dimensions. The entire architecture is length independent, which allows us to process trajectories of arbitrary lengths.

Convolutional module – The first main convolutional module allows us to expand the trajectory dimension with several convolutional filters. This provides the following layers with a richer embedding based on short-range correlations.

We build it following the XResNet [61] architecture. As we show in Fig. A.1, it consists of an initial convolutional layer, commonly referred to as the *stem*, followed by a series of *residual blocks* that feature a convolutional layer with a skip connection. We use one-dimensional convolutions with a kernel size of three and stride one to preserve the trajectory size. However, we use a kernel size of one in the skip connections, which can act as the identity or a scaling factor whose main purpose is to match the tensor shapes on both paths of the residual blocks, as we explain below. This module can take a batch of input trajectories of size $[\texttt{batch\_size} \times T \times d]$ and output a batch of features of size $[\texttt{batch\_size} \times T \times \texttt{embedding\_size}]$.

Throughout the architecture, we add a batch normalization layer directly after every convolutional layer, and we use the rectified linear unit (ReLU) activation function by default, except in the last output layer.

To produce the results, we use a single convolutional layer and a ReLU activation in the stem. We use 64 filters to predict the diffusion coefficient and 32 filters for the

anomalous diffusion exponent. Then, we have added three residual blocks with 128, 256 and 512 filters, respectively. Hence, `embedding_size` $= 512$. In these blocks, the convolutional paths have two convolutional layers: the first one increases the embedding size and the second one preserves the dimensions. In the skip connection, we only have one convolutional layer that increases the embedding size to match the dimensions of the convolutional path. We implement the ReLU activation at the end of the block, after we add the outcome of both paths.

Self-attention module – We process the features extracted by the convolutional module with a self-attention mechanism that allows the model to capture long-range correlations.

More precisely, we implement a *transformer encoder* [66], as we explain in Section 2.2.2. As we illustrate in Fig. A.1, the encoder block has two main parts, both featuring a skip connection followed by a layer normalization after the sum of both paths. In the first one, we have a multi-head attention layer that feeds on the input and, in the second one, we have a couple of pointwise feedforward layers, which are equally applied to each element in the incoming tensor. Furthermore, we can add a positional encoding before the first encoder block, which provides information about the relative position of each element in the trajectory. This module can process a batch of embeddings preserving its dimensions. Hence, the input and the output both have size $[\texttt{batch\_size} \times T \times \texttt{embedding\_size}]$.

To produce the results, we use four transformer encoder blocks with eight heads in the multi-head attention layers. The pointwise feedforward part adds two fully-connected layers with `embedding_size` neurons each, i.e, 512 in this case. Interestingly, we have found that, after the convolutions, the positional encoding has very little impact on the results. Therefore, in the interest of simplicity, we have not used it to obtain the results reported in this work.

Feedforward module – The last main component is a shallow feedforward fully-connected network that acts element-wise on the features extracted by the previous module. We tailor this part to the specific task at hand to achieve the desired output with the proper dimensions.

For instance, in a regression task, the output dimension is one and we use a scaled sigmoid activation function at the end to define the output range with some margin, e.g., $\log D \in (-3.1, 3.1)$, $\alpha \in (0, 2.05)$. This margin allows the sigmoid to reach the desired values before it saturates. In a hypothetical case of classification task (as e.g. classifying between diffusion models as done in Ref. [367]), the final dimension is the number of classes and we use a softmax activation function. Then, we obtain the predictions by choosing the class with the maximum activation value. Hence, we can process a feature batch of size $[\texttt{batch\_size} \times T \times \texttt{embedding\_size}]$ and output their predictions with size $[\texttt{batch\_size} \times T \times \texttt{num\_class}]$. In case that $\texttt{num\_class} > 1$, as in a classification task, we perform an additional post-processing step to obtain an output of size $[\texttt{batch\_size} \times T \times 1]$ with the corresponding predictions at each time step.

A.2 Data set details

Throughout Chapter 8, we present a series of results that rely on different sets of data to evaluate different aspects of STEP. In Table A.1, we provide all the details about the data sets used to train and generate all the presented results.

There, the $[\,]$ denote closed ranges, and $\{\}$ denote sets. The values for D and σ_{noise} are in $\log_{10}$ scale, and we take α intervals of 0.05 within the denoted ranges.

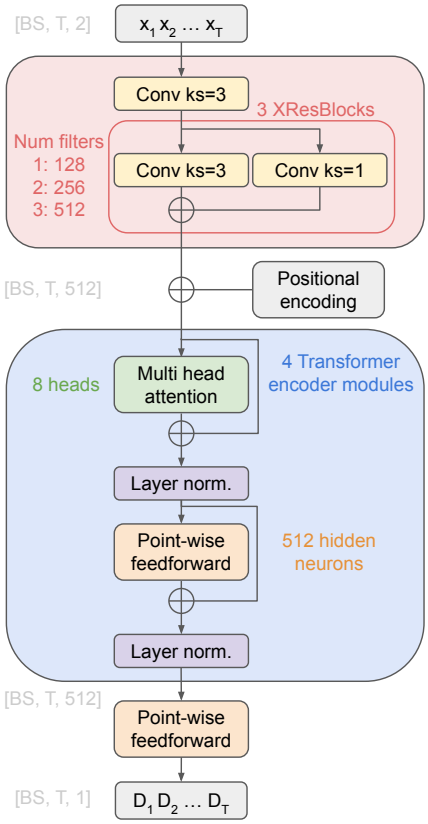

FIGURE A.1: Machine learning architecture representation

More precisely, we consider different α ranges for every anomalous diffusion model. We simulate annealed transit-time model (ATTM) [341] with $\alpha \in [0.05, 1]$, continuous time random walk (CTRW) [337] with $\alpha \in [0.05, 1]$, fractional Brownian motion (FBM) [343] with $\alpha \in [0.05, 1.95]$, Lévy walk (LW) [339] with $\alpha \in [1.05, 2]$, and scaled Brownian motion (SBM) [347] with $\alpha \in [0.05, 2]$. The ranges of D are logarithmically equispaced and we take 1000 unique values unless stated otherwise. All the data sets with 2 to 5 segments have their lengths sampled according to an exponential distribution with a minimum length of 10 and a maximum of 190 steps, with an average of ~ 57 time steps. All the data is simulated, except for the two experimental data sets denoted with "(exp)" from which we do not know the ground truth.

Task	Models	D	α	σ_{noise}	Traj. len.	# segs.	Seg. len.	Size
Train BM	Brownian motion	$[-3, 3]$	1	$[-6, 2]$	200	$[2, 5]$	$[10, 190]$	100,000
Train AnDi	All anomalous	1	$[0.05, 2]$	0.1	200	$[2, 5]$	$[10, 190]$	100,064
Fig. 8.2(a), (b); Fig. 8.3(g)	Brownian motion	$[-3, 3]$	1	0	200	$[2, 5]$	$[10, 190]$	48,000
Fig. 8.2(c); Fig. 8.3(e), (f)	All anomalous	1	$[0.05, 2]$	$[-5, 2]$	200	$[2, 5]$	$[10, 190]$	49,994
Fig. 8.2(d); Fig. 8.3(d)	All anomalous	1	$[0.05, 2]$	$\{0, 0.1\}$	200	$[2, 5]$	$[10, 190]$	50,000
Fig. 8.2(e), (f)	Brownian motion	$[-3, 3]$	1	0	200	2	$[10, 190]$	50,000
Fig. 8.2(g), (h)	FBM	1	$[0.05, 1.95]$	0	200	2	$[10, 190]$	40,000
Fig. 8.4	SBM	1	$\{0.1, 0.5\}$	0	200	1	200	6,000
Fig. 8.5(b), (c), (d)	ATTM	(-6.7, 0)	0.75	0	200	$[1, 51]$	$[1, 200]$	10,000
Fig. 8.5(e), (f), (g)	(exp)				$[200, 2000]$			755
Fig. 8.6	(exp)				$[20, 500]$			4734
Fig. 8.3(a)	Brownian motion	$[-3, 3]$	1	0	200	[2, 5]	[10, 190]	200,384
Fig. 8.3(b)	Brownian motion	$[-3, 3]$	1	$[-6, 0]$	200	$[2, 5]$	$[10, 190]$	48,000
Fig. 8.3(c)	Brownian motion	$[-3, 3]$	1	0	$[40, 660]$	$[2, 11]$	$\{20, 40, 60\}$	20,000
Fig. 8.3(h)	Brownian motion	$[-3, 3]$	1	0	$[20, 660]$	$[1, 11]$	$\{20, 40, 60\}$	22,000

TABLE A.1: **Data set details for all the results reported in Chapter 8.** We use 20% of the training data (first two rows) for validation and hyperparameter tuning, whereas the rest is used for testing. We use the same Brownian motion test set from Fig. 8.2(a) & (b) to predict α for Fig. 8.3(g). We take two independent sub-samples of a test set with 199,976 trajectories: one for Fig. 8.2(c), Fig. 8.3(e) & (f), and the other for Fig. 8.2(d), Fig. 8.3(d). In Fig. 8.2(d), we consider noiseless trajectories, although we add noise with $\sigma_{\text{noise}} = 0.1$ to flat CTRW segments that would result in numerical instabilities for the TA-MSD method. To generate the noisy trajectories for Fig. 8.2(c), and Fig. 8.3(e) & (f) we add 128 random levels of localization noise to each trajectory in the data set, effectively making about 6.4×10^6 trajectories. In Fig. 8.4, we evenly split the two values of α among all trajectories. In Fig. 8.5(b), (c) & (d) we use ATTM trajectories with $\sigma = 0.3$ and $\gamma = 0.4$ (see [324, 341], do not confuse with σ_{noise}). We simulate them randomly sampling D and the segment lengths accordingly, and the values here come from analysing the resulting trajectories, which have 18 different segments on average. For Fig. 8.5(e), (f) & (g) we consider 755 experimental trajectories of the pathogen-recognition receptor DC-SIGN containing from 200 to 2000 frames sampled at 60 Hz. We obtain all the results in Fig. 8.6 from 4734 trajectories of the integrin $\alpha 5 \beta 1$ containing from 20 to 500 frames sampled at 33 Hz. In Fig. 8.3(a), we average over all D values for each segment length to estimate the uncertainty. We take D in intervals of 0.2 within the given range (31 values), which combined with over 2×10^5 trajectories (over 7×10^5 segments combined) result in $\sim$ 125 segments per D and length value. In Fig. 8.3(b), we use the same trajectories from Fig. 8.2(a) & (b) and add 128 random levels of localization noise to each of them, effectively making 6.144×10^6 noisy trajectories. The data set for Fig. 8.3(h) is a sub-set of the one from Fig. 8.3(c).

List of Abbreviations

AI artificial intelligence

ATTM annealed transit-time model

BFS breadth first search

CNN convolutional neural network

CR cross-resonance

CRLB Cramér-Rao lower bound

CTRW continuous time random walk

DL deep learning

DQN deep Q-network

EA-MSD ensemble-averaged MSD

FBM fractional Brownian motion

GPT generative pre-trained transformer

GPU graphics processing unit

GRU gated recurrent unit

HVG horizontal visibility graph

HVGM horizontal visibility graph motif

JI Jaccard index

KCPD kernel changepoint detection

LR logistic regression

LSTM long short-term memory

LW Lévy walk

MAE mean absolute error

MC Monte Carlo

MDP Markov decision process

ML machine learning

MSD mean squared displacement

NISQ noisy intermediate-scale quantum

NLP natural language processing

NN neural network

NQS neural quantum state

PCA principal component analysis

PPT positive under partial transposition

PWC piecewise constant

QMP quantum marginal problem

RDM reduced density matrix

RF random forest

RL reinforcement learning

RNN recurrent neural network

SBM scaled Brownian motion

SdP semidefinite programming

SVM support vector machine

TA-MSD time-averaged MSD

TD temporal-difference

TL transfer learning

UMAP uniform manifold approximation and projection

XGB extreme gradient boosting

List of Figures

List of Tables

Bibliography

[1] "December 1958: Invention of the Laser". *APS News* 12.11 (2003).

[2] Albert Einstein. "Zur Quantentheorie der Strahlung". *Phys Zeit* 18, p. 121 (1917).

[3] B. P. Abbott et al. "Observation of Gravitational Waves from a Binary Black Hole Merger". *Phys. Rev. Lett.* 116 (6), p. 061102 (2016).

[4] Ferenc Krausz. "The birth of attosecond physics and its coming of age". *Physica Scripta* 91.6, p. 063011 (2016).

[5] Reiner Weiss, Barry C. Barish, and Kip S. Thorne. *Nobel prize in physics 2017*.

[6] Pierre Agostini, Ferenc Krausz, and Anne L'Huillier. *Nobel prize in physics 2023*.

[7] W.F. Brinkman, D.E. Haggan, and W.W. Troutman. "A history of the invention of the transistor and where it will lead us". *IEEE Journal of Solid-State Circuits* 32.12, pp. 1858–1865 (1997).

[8] Thierry Dauxois. "Fermi, Pasta, Ulam, and a mysterious lady". *Physics Today* 61.1, pp. 55–57 (2008). ISSN: 0031-9228.

[9] Hanchen Wang et al. "Scientific discovery in the age of artificial intelligence". *Nature* 620.79727972, pp. 47–60 (2023). ISSN: 1476-4687.

[10] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. "Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics". *Phys. Rev. Lett.* 120 (14), p. 143001 (2018).

[11] Remi Lam et al. "Learning skillful medium-range global weather forecasting". *Science* 382.6677, pp. 1416–1421 (2023).

[12] The Event Horizon Telescope Collaboration et al. "First M87 Event Horizon Telescope Results. IV. Imaging the Central Supermassive Black Hole". *The Astrophysical Journal Letters* 875.1, p. L4 (2019).

[13] Maxwell L. Bileschi et al. "Using deep learning to annotate the protein universe". *Nature Biotechnology* 40.6, pp. 932–937 (2022). ISSN: 1546-1696.

[14] Regine S Bohacek, Colin McMartin, and Wayne C Guida. "The art and practice of structure-based drug design: a molecular modeling perspective". *Med. Res. Rev.* 16.1, pp. 3–50 (1996).

[15] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. "Inverse molecular design using machine learning: Generative models for matter engineering". *Science* 361.6400, pp. 360–365 (2018).

[16] Richard D. Ball et al. "Evidence for intrinsic charm quarks in the proton". *Nature* 608.7923, pp. 483–487 (2022). ISSN: 1476-4687.

[17] Alex Davies et al. "Advancing mathematics by guiding human intuition with AI". *Nature* 600.7887, pp. 70–74 (2021). ISSN: 1476-4687.

[18] Adam Zsolt Wagner. *Constructions in combinatorics via neural networks*. 2021. arXiv: 2104.14516.

[19] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". *Nature* 596.7873, pp. 583–589 (2021).

[20] Mihaly Varadi et al. "AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models". *Nucleic Acids Res.* 50.D1, pp. D439–D444 (2021). ISSN: 0305-1048.

[21] Connor W. Coley et al. "A robotic platform for flow synthesis of organic compounds informed by AI planning". *Science* 365.6453, eaax1566 (2019).

[22] Alexey A Melnikov et al. "Active learning machine learns to create new quantum experiments". *Proc. Natl. Acad. Sci. U.S.A.* 115.6, pp. 1221–1226 (2018).

[23] Jonas Degrave et al. "Magnetic control of tokamak plasmas through deep reinforcement learning". *Nature* 602.7897, pp. 414–419 (2022). ISSN: 1476-4687.

[24] Tony Hey, Stewart Tansley, Kristin Tolle, and Jim Gray. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research (2009).

[25] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih. "Machine learning in the search for new fundamental physics". *Nature Reviews Physics* 4.6, pp. 399–412 (2022). ISSN: 2522-5820.

[26] Vito P. Pastore, Thomas G. Zimmerman, Sujoy K. Biswas, and Simone Bianco. "Annotation-free learning of plankton for classification and anomaly detection". *Scientific Reports* 10.1, p. 12142 (2020). ISSN: 2045-2322.

[27] Mario Krenn and Anton Zeilinger. "Predicting research trends with semantic and neural networks with an application in quantum physics". *Proceedings of the National Academy of Sciences* 117.4, pp. 1910–1916 (2020).

[28] Qing Ke, Alexander J. Gates, and Albert-László Barabási. "A network-based normalized impact measure reveals successful periods of scientific discovery across discipline". *Proceedings of the National Academy of Sciences* 120.48, e2309378120 (2023).

[29] Christopher Bishop. "AI4Science to empower the fifth paradigm of scientific discovery". *Microsoft Research Blog* (2022).

[30] Leon Klein et al. "Timewarp: Transferable Acceleration of Molecular Dynamics by Learning Time-Coarsened Dynamics". *NeurIPS 2023 - Adv. Neural Inf. Process. Syst.* (2023).

[31] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". *Science* 362.6419, pp. 1140–1144 (2018).

[32] Alhussein Fawzi et al. "Discovering faster matrix multiplication algorithms with reinforcement learning". *Nature* 610.7930, pp. 47–53 (2022). ISSN: 1476-4687.

[33] Daniel J. Mankowitz et al. "Faster sorting algorithms discovered using deep reinforcement learning". *Nature* 618.7964, pp. 257–263 (2023).

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186 (2019).

[35] Noelia Ferruz and Birte Höcker. "Controllable protein design with language models". *Nature Machine Intelligence* 4.6, pp. 521–532 (2022). ISSN: 2522-5839.

[36] Brian Hie, Ellen D. Zhong, Bonnie Berger, and Bryan Bryson. "Learning the language of viral evolution and escape". *Science* 371.6526, pp. 284–288 (2021).

[37] Anna Dawid et al. *Modern applications of machine learning in quantum sciences*. 2022. arXiv: 2204.04198.

[38] Borja Requena, Marcin Płodzień, Paolo Stornati, and Alexandre Dauphin. *BorjaRequena/Neural-Network-Course*. GitHub (2022).

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press (2016). http://www.deeplearningbook.org.

[40] Tom M Mitchell. *Machine learning*. McGraw-Hill, New York (1997).

[41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". *Proc. IEEE* 86.11, pp. 2278–2324 (1998).

[42] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. MIT & NYU, 2009.

[43] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". *Int. J. Comput. Vis.* 115.3, pp. 211–252 (2015).

[44] Borja Requena. *BorjaRequena/GPTutorial*. GitHub (2023).

[45] Jonathan Frankle and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. arXiv: 1803.03635 [cs.LG].

[46] Stephen Hanson and Lorien Pratt. "Comparing Biases for Minimal Network Construction with Back-Propagation". *Advances in Neural Information Processing Systems* (1988).

[47] Anders Krogh and John Hertz. "A Simple Weight Decay Can Improve Generalization". *Advances in Neural Information Processing Systems* (1991).

[48] William Finnoff, Ferdinand Hergert, and Hans Georg Zimmermann. "Improving model selection by nonconvergent methods". *Neural Networks* 6.6, pp. 771–783 (1993). ISSN: 0893-6080.

[49] P.Y. Simard, D. Steinkraus, and J.C. Platt. "Best practices for convolutional neural networks applied to visual document analysis". *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Pp. 958–963 (2003).

[50] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". *CoRR* abs/1207.0580 (2012). arXiv: 1207.0580.

[51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". *Nature* 521.7553, pp. 436–444 (2015). ISSN: 1476-4687.

[52] Andrei Nikolaevich Kolmogorov. "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition". *Doklady Akademii Nauk*, pp. 953–956 (1957).

[53] George Cybenko. "Approximation by superpositions of a sigmoidal function". *Math. Control Signals Syst.* 2.4, pp. 303–314 (1989).

[54] Uriel Singer et al. "Make-A-Video: Text-to-Video Generation without Text-Video Data". *The Eleventh International Conference on Learning Representations* (2023).

[55] James Betker et al. “Improving image generation with better captions”. *Computer Science* (2023).

[56] Evelyn Fix and J. L. Hodges. “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. *International Statistical Review / Revue Internationale de Statistique* 57.3, pp. 238–247 (1989). ISSN: 03067734, 17515823.

[57] T. Cover and P. Hart. “Nearest neighbor pattern classification”. *IEEE Transactions on Information Theory* 13.1, pp. 21–27 (1967).

[58] Leo Breiman. “Random Forests”. *Machine Learning* 45.1, pp. 5–32 (2001). ISSN: 1573-0565.

[59] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. *International Conference on Learning Representations* (2014).

[60] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. *Neural Computation* 1.4, pp. 541–551 (1989).

[61] Tong He et al. “Bag of tricks for image classification with convolutional neural networks”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 558–567 (2019).

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (2016).

[63] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. *Neural Computation* 9.8, pp. 1735–1780 (1997).

[64] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734 (2014).

[65] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. *NeurIPS 2014 Workshop on Deep Learning* (2014).

[66] Ashish Vaswani et al. “Attention is All you Need”. *Advances in Neural Information Processing Systems* (2017).

[67] Tom Brown et al. “Language Models are Few-Shot Learners”. *Advances in Neural Information Processing Systems*, pp. 1877–1901 (2020).

[68] S. Richard Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Bradford Book (2018).

[69] Richard S Sutton. “Learning to predict by the methods of temporal differences”. *Mach. Learn.* 3 (1), pp. 9–44 (1988).

[70] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. CUED/F-INFENG/TR 166. University of Cambridge, Department of Engineering, 1994.

[71] Harm van Seijen, A Rupam Mahmood, Patrick M Pilarski, Marlos C Machado, and Richard S Sutton. “True Online Temporal-Difference Learning”. *J. Mach. Learn. Res.* 17 (145), pp. 1–40 (2016).

[72] George H. John. “When the Best Move Isn’t Optimal: Q-Learning with Exploration”. *Proc. 12th Nat. Conf. Artif. Intell. (Vol. 2)* (1994).

[73] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. *Mach. Learn.* 8 (3), pp. 279–292 (1992).

[74] James E Smith and Robert L Winkler. "The optimizer's curse: Skepticism and postdecision surprise in decision analysis". *Manag. Sci.* 52 (3), pp. 311–322 (2006).

[75] Sebastian Thrun and Anton Schwartz. "Issues in using function approximation for reinforcement learning". *Proc. 4th Connectionist Models Summer School* (1993).

[76] Hado Van Hasselt. "Double Q-learning". *NIPS 2010 - Adv. Neural Inf. Process. Syst.* (2010).

[77] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". *Nature* 518.7540, pp. 529–533 (2015).

[78] Long-Ji Lin. "Reinforcement Learning for Robots Using Neural Networks". UMI Order No. GAX93-22750. PhD thesis. USA: Carnegie Mellon University, 1992.

[79] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double Q-learning". *Proc. AAAI Conf. Artif. Intell.* (2016). arXiv: 1509.06461.

[80] Peter Marbach and John N Tsitsiklis. "Simulation-based optimization of Markov reward processes". *IEEE Trans. Automat. Contr.* 46 (2), pp. 191–209 (2001).

[81] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". *NIPS 1999 - Adv. Neural Inf. Process. Syst.* (1999).

[82] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". *Mach. Learn.* 8 (3), pp. 229–256 (1992). ISSN: 1573-0565.

[83] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. "Self-Critical Sequence Training for Image Captioning". *Proc. IEEE Conf. Comput. Vision and Pattern Recognition* (2017).

[84] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". *ICLR 2016 - Int. Conf. Learn. Represent.* (2016). arXiv: 1506.02438.

[85] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". *IEEE Trans. Syst. Man Cybern. Syst.* 5, pp. 834–846 (1983).

[86] Vijay Konda and John Tsitsiklis. "Actor-Critic Algorithms". *Adv. Neural Inf. Process. Syst.* (1999).

[87] Thomas Degris, Martha White, and Richard S. Sutton. "Off-Policy Actor-Critic". *ICML 2012 - Conf. Mach. Learn.* Pp. 179–186 (2012). eprint: 1205.4839.

[88] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". *ICML 2016 - 33th Int. Conf. Mach. Learn.* Pp. 1928–1937 (2016). eprint: 1602.01783.

[89] Shun-ichi Amari. "Natural Gradient Works Efficiently in Learning". *Neural Comput.* 10.2, pp. 251–276 (1998).

[90] Sham M Kakade. "A Natural Policy Gradient". *NIPS 2001 - Adv. Neural Inf. Process. Syst.* (2001).

[91] Jan Peters and Stefan Schaal. "Natural Actor-Critic". *Neurocomputing* 71.7, pp. 1180–1190 (2008).

[92] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. "Natural actor–critic algorithms". *Automatica* 45.11, pp. 2471–2482 (2009).

[93] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust Region Policy Optimization". *ICML 2015 - Int. Conf. Mach. Learn.* (2015). arXiv: 1502.05477.

[94] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation". *NIPS 2017 - Adv. Neural Inf. Process. Syst.* (2017). arXiv: 1708.05144.

[95] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[96] Paul Adrien Maurice Dirac and Ralph Howard Fowler. "Quantum mechanics of many-electron systems". *Proc. R. Soc. A: Math. Phys. Eng. Sci.* 123.792, pp. 714–733 (1929).

[97] R. P. Feynman. "Simulating physics with computers". *Int. J. Theor. Phys.* 21.6-7 (1982).

[98] Markus Greiner, Olaf Mandel, Tilman Esslinger, Theodor W. Hänsch, and Immanuel Bloch. "Quantum phase transition from a superfluid to a Mott insulator in a gas of ultracold atoms". *Nature* 415.6867, pp. 39–44 (2002). ISSN: 1476-4687.

[99] B. Keimer, S. A. Kivelson, M. R. Norman, S. Uchida, and J. Zaanen. "From quantum matter to high-temperature superconductivity in copper oxides". *Nature* 518.7538, pp. 179–186 (2015). ISSN: 1476-4687.

[100] S. Lloyd. "Universal quantum simulators". *Science* 273.5278, pp. 1073–1078 (1996).

[101] Giuseppe Carleo et al. "Machine learning and the physical sciences". *Rev. Mod. Phys.* 91 (4), p. 045002 (2019).

[102] Juan Carrasquilla. "Machine learning for quantum matter". *Advances in Physics: X* 5.1, p. 1797528 (2020).

[103] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press (2011).

[104] Juan Carrasquilla and Roger G Melko. "Machine learning phases of matter". *Nat. Phys.* 13.5, pp. 431–434 (2017).

[105] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. "Learning phase transitions by confusion". *Nature Physics* 13.5, pp. 435–439 (2017). ISSN: 1745-2481.

[106] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. "Identifying quantum phase transitions with adversarial neural networks". *Phys. Rev. B* 97 (13), p. 134109 (2018).

[107] Ye-Hua Liu and Evert P. L. van Nieuwenburg. "Discriminative Cooperative Networks for Detecting Phase Transitions". *Phys. Rev. Lett.* 120 (17), p. 176401 (2018).

[108] Korbinian Kottmann, Patrick Huembeli, Maciej Lewenstein, and Antonio Acín. "Unsupervised Phase Discovery with Deep Anomaly Detection". *Phys. Rev. Lett.* 125 (17), p. 170603 (2020).

[109] Anna Dawid, Patrick Huembeli, Michal Tomza, Maciej Lewenstein, and Alexandre Dauphin. "Phase detection with neural networks: interpreting the black box". *New Journal of Physics* 22.11, p. 115001 (2020).

[110] Annabelle Bohrdt et al. "Classifying snapshots of the doped Hubbard model with machine learning". *Nat. Phys.* 15.9, pp. 921–924 (2019). ISSN: 17452481.

[111] B S Rem et al. "Identifying quantum phase transitions using artificial neural networks on experimental data". *Nat. Phys.* 15, pp. 917–920 (2019).

[112] Niklas Käming et al. "Unsupervised machine learning of topological phase transitions from experimental data". *Mach. Learn.: Sci. Technol.* 2, p. 035037 (2021).

[113] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". *Science* 355.6325, pp. 602–606 (2017).

[114] Kenny Choo, Giuseppe Carleo, Nicolas Regnault, and Titus Neupert. "Symmetries and Many-Body Excitations with Neural-Network Quantum States". *Phys. Rev. Lett.* 121 (16), p. 167204 (2018).

[115] Xiao Liang et al. "Solving frustrated quantum many-particle models with convolutional neural networks". *Phys. Rev. B* 98 (10), p. 104426 (2018).

[116] Mohamed Hibat-Allah, Martin Ganahl, Lauren E Hayward, Roger G Melko, and Juan Carrasquilla. "Recurrent neural network wave functions". *Phys. Rev. Res.* 2.2, p. 023358 (2020).

[117] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. "Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems". *Phys. Rev. Lett.* 124.2, p. 020503 (2020). ISSN: 1079-7114.

[118] Agnes Valenti, Eliska Greplova, Netanel H. Lindner, and Sebastian D. Huber. "Correlation-enhanced neural networks as interpretable variational quantum states". *Phys. Rev. Res.* 4 (1), p. L012010 (2022).

[119] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. "Quantum Entanglement in Neural Network States". *Phys. Rev. X* 7 (2), p. 021021 (2017).

[120] Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. "Quantum Entanglement in Deep Learning Architectures". *Phys. Rev. Lett.* 122 (6), p. 065301 (2019).

[121] Or Sharir, Amnon Shashua, and Giuseppe Carleo. "Neural tensor contractions and the expressive power of deep neural quantum states". *Phys. Rev. B* 106 (20), p. 205136 (2022).

[122] Román Orús. "A practical introduction to tensor networks: Matrix product states and projected entangled pair states". *Ann. Phys. (N. Y.)* 349, pp. 117–158 (2014). ISSN: 0003-4916.

[123] Yusuke Nomura, Andrew S. Darmawan, Youhei Yamaji, and Masatoshi Imada. "Restricted Boltzmann machine learning for solving strongly correlated quantum systems". *Phys. Rev. B* 96 (20), p. 205152 (2017).

[124] Kenny Choo, Titus Neupert, and Giuseppe Carleo. "Two-dimensional frustrated $J_1 - J_2$ model studied with neural network quantum states". *Phys. Rev. B* 100.12 (2019). ISSN: 2469-9969.

[125] Markus Schmitt and Markus Heyl. "Quantum Many-Body Dynamics in Two Dimensions with Artificial Neural Networks". *Phys. Rev. Lett.* 125 (10), p. 100503 (2020).

[126] Jan Hermann et al. "Ab initio quantum chemistry with neural-network wavefunctions". *Nat. Rev. Chem.* 7.10, pp. 692–709 (2023). ISSN: 2397-3358.

[127] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. "Ab initio solution of the many-electron Schrödinger equation with deep neural networks". *Phys. Rev. Res.* 2 (3), p. 033429 (2020).

[128] Gino Cassella et al. "Discovering Quantum Phase Transitions with Fermionic Neural Networks". *Phys. Rev. Lett.* 130 (3), p. 036401 (2023).

[129] Jan Hermann, Zeno Schätzle, and Frank Noé. "Deep-neural-network solution of the electronic Schrödinger equation". *Nat. Chem.* 12.10, pp. 891–897 (2020). ISSN: 1755-4349.

[130] Borja Requena. *Introduction to variational Monte Carlo with neural network quantum states*. 2021.

[131] Federico Becca and Sandro Sorella. *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press (2017).

[132] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. "Self-learning Monte Carlo method". *Phys. Rev. B* 95 (4), p. 041101 (2017).

[133] Li Huang and Lei Wang. "Accelerated Monte Carlo simulations with restricted Boltzmann machines". *Phys. Rev. B* 95 (3), p. 035105 (2017).

[134] Troels Arnfred Bojesen. "Policy-guided Monte Carlo: Reinforcement-learning Markov chain dynamics". *Phys. Rev. E* 98 (6), p. 063303 (2018).

[135] Kai-Wen Zhao, Wen-Han Kao, Kai-Hsin Wu, and Ying-Jer Kao. "Generation of ice states through deep reinforcement learning". *Phys. Rev. E* 99 (6), p. 062106 (2019).

[136] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning". *Science* 365.6457, eaaw1147 (2019).

[137] Kim A. Nicoli et al. "Asymptotically unbiased estimation of physical observables with neural samplers". *Phys. Rev. E* 101 (2), p. 023304 (2020).

[138] Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, and Juan Carrasquilla. "Variational neural annealing". *Nat. Mach. Intell.* 3 (11), pp. 952–961 (2021). ISSN: 2522-5839.

[139] Ted Moskovitz, Michael Arbel, Ferenc Huszar, and Arthur Gretton. "Efficient Wasserstein Natural Gradients for Reinforcement Learning". *International Conference on Learning Representations* (2021).

[140] Kirill Neklyudov et al. "Wasserstein Quantum Monte Carlo: A Novel Approach for Solving the Quantum Many-Body Schrödinger Equation". *Advances in Neural Information Processing Systems*, pp. 63461–63482 (2023).

[141] Dongdong Wang et al. "Efficient sampling of high-dimensional free energy landscapes using adaptive reinforced dynamics". *Nat. Comput. Sci.* 2.1, pp. 20–29 (2022). ISSN: 2662-8457.

[142] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost". *Chemical science* 8.4, pp. 3192–3203 (2017).

[143] Manuel Erhard, Mario Krenn, and Anton Zeilinger. "Advances in high-dimensional quantum entanglement". *Nat. Rev. Phys.* 2.7, pp. 365–381 (2020).

[144] Mario Krenn, Manuel Erhard, and Anton Zeilinger. "Computer-inspired quantum experiments". *Nat. Rev. Phys.* 2, pp. 649–661 (2020).

[145] Mario Krenn, Mehul Malik, Robert Fickler, Radek Lapkiewicz, and Anton Zeilinger. "Automated search for new quantum experiments". *Phys. Rev. Lett.* 116 (9), p. 090405 (2016).

[146] Mario Krenn, Jakob S. Kottmann, Nora Tischler, and Alán Aspuru-Guzik. "Conceptual understanding through efficient automated design of quantum optical experiments". *Phys. Rev. X* 11 (3), p. 031044 (2021).

[147] Carla Rodríguez, Sören Arlt, Leonhard Möckl, and Mario Krenn. *XLuminA: An Auto-differentiating Discovery Framework for Super-Resolution Microscopy*. 2023. arXiv: 2310.08408 [physics.optics].

[148] Florian Häse, Loïc M. Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. "Phoenics: A Bayesian optimizer for chemistry". *ACS Cent. Sci.* 4.9, pp. 1134–1145 (2018).

[149] Marin Bukov et al. "Reinforcement Learning in Different Phases of Quantum Control". *Phys. Rev. X* 8 (3), p. 031086 (2018).

[150] J. Duris et al. "Bayesian optimization of a free-electron laser". *Phys. Rev. Lett.* 124 (12), p. 124801 (2020).

[151] Eliska Greplova et al. "Fully automated identification of two-dimensional material samples". *Phys. Rev. Appl.* 13.6, p. 064017 (2020).

[152] Renato Durrer et al. "Automated tuning of double quantum dots into specific charge states using neural networks". *Phys. Rev. Appl.* 13.5, p. 054019 (2020).

[153] Kevin Reuer et al. "Realizing a deep reinforcement learning agent for real-time quantum feedback". *Nat. Commun.* 14.1, p. 7138 (2023). ISSN: 2041-1723.

[154] P. B. Wigley et al. "Fast machine-learning online optimization of ultra-cold-atom experiments". *Sci. Rep.* 6.1, p. 25890 (2016). ISSN: 2045-2322.

[155] Anna Dawid, Niccolò Bigagli, Daniel W. Savin, and Sebastian Will. *Automated detection of laser cooling schemes for ultracold molecules*. 2023. arXiv: 2311.08381 [quant-ph].

[156] Giacomo Torlai et al. "Neural-network quantum state tomography". *Nat. Phys.* 14 (5), pp. 447–450 (2018). ISSN: 1745-2481.

[157] Giacomo Torlai et al. "Integrating Neural Networks with a Quantum Simulator for State Reconstruction". *Phys. Rev. Lett.* 123 (23), p. 230504 (2019).

[158] Tobias Schmale, Moritz Reh, and Martin Gärttner. "Efficient quantum state tomography with convolutional neural networks". *npj Quantum Information* 8.1 (2022).

[159] James R. Garrison and Tarun Grover. "Does a Single Eigenstate Encode the Full Hamiltonian?" *Phys. Rev. X* 8 (2), p. 021026 (2018).

[160] Eyal Bairey, Itai Arad, and Netanel H. Lindner. "Learning a Local Hamiltonian from Local Measurements". *Phys. Rev. Lett.* 122 (2), p. 020504 (2019).

[161] Agnes Valenti, Guliuxin Jin, Julian Léonard, Sebastian D. Huber, and Eliska Greplova. "Scalable Hamiltonian learning for large-scale out-of-equilibrium quantum dynamics". *Phys. Rev. A* 105 (2), p. 023302 (2022).

[162] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge university press (2010).

[163] L. Hales and S. Hallgren. "An improved quantum Fourier transform algorithm and applications". *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 515–525 (2000).

[164] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134 (1994).

[165] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219 (1996).

[166] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". *Phys. Rev. Lett.* 103 (15), p. 150502 (2009).

[167] John Preskill. "Quantum computing in the NISQ era and beyond". *Quantum* 2, p. 79 (2018). ISSN: 2521-327X.

[168] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". *Nat. Commun.* 5.1 (2014). ISSN: 2041-1723.

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

[170] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. "Evaluating analytic gradients on quantum hardware". *Phys. Rev. A* 99 (3), p. 032331 (2019).

[171] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. "Barren plateaus in quantum neural network training landscapes". *Nat. Commun.* 9.1, pp. 1–6 (2018).

[172] Joseph Bowles, David Wierichs, and Chae-Yeun Park. *Backpropagation scaling in parameterised quantum circuits*. 2023. arXiv: 2306.14962 [quant-ph].

[173] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. "Quantum Natural Gradient". *Quantum* 4, p. 269 (2020). ISSN: 2521-327X.

[174] Patrick Huembeli and Alexandre Dauphin. "Characterizing the loss landscape of variational quantum circuits". *Quantum Science and Technology* 6.2, p. 025011 (2021).

[175] Jiahao Yao, Marin Bukov, and Lin Lin. "Policy Gradient based Quantum Approximate Optimization Algorithm". *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, pp. 605–634 (2020).

[176] Guillaume Verdon et al. *Learning to learn with quantum neural networks via classical neural networks*. 2019. arXiv: 1907.05415.

[177] Friederike Metz and Marin Bukov. "Self-correcting quantum many-body control using reinforcement learning with tensor networks". *Nat. Mach. Intell.* 5.7, pp. 780–791 (2023). ISSN: 2522-5839.

[178] V. V. Sivak et al. "Model-Free Quantum Control with Reinforcement Learning". *Phys. Rev. X* 12 (1), p. 011059 (2022).

[179] Jeongho Bang, Junghee Ryu, Seokwon Yoo, Marcin Pawłowski, and Jinhyoung Lee. "A strategy for quantum algorithm design assisted by machine learning". *New Journal of Physics* 16.7, p. 073017 (2014).

[180] Lea M Trenkwalder, Andrea López-Incera, Hendrik Poulsen Nautrup, Fulvio Flamini, and Hans J Briegel. "Automated gadget discovery in the quantum domain". *Machine Learning: Science and Technology* 4.3, p. 035043 (2023).

[181] Thomas Fösel, Murphy Yuezhen Niu, Florian Marquardt, and Li Li. *Quantum circuit optimization with deep reinforcement learning*. 2021. arXiv: 2103.07585 [quant-ph].

[182] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: 1811.04968 [quant-ph].

[183] Gadi Aleksandrowicz et al. *Qiskit: An Open-source Framework for Quantum Computing*. Zenodo (2019).

[184] Bjarni Jonsson, Bela Bauer, and Giuseppe Carleo. *Neural-network states for the classical simulation of quantum computing*. 2018. arXiv: 1808.05232 [quant-ph].

[185] Matija Medvidović and Giuseppe Carleo. "Classical variational simulation of the Quantum Approximate Optimization Algorithm". *npj Quantum Inf.* 7 (1), p. 101 (2021). ISSN: 2056-6387.

[186] Florian Fürrutter, Gorka Muñoz-Gil, and Hans J. Briegel. *Quantum circuit synthesis with diffusion models*. 2023. arXiv: 2311.02041 [quant-ph].

[187] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". *Nature* 574.7779, pp. 505–510 (2019).

[188] Han-Sen Zhong et al. "Quantum computational advantage using photons". *Science* 370.6523, pp. 1460–1463 (2020).

[189] Lars S. Madsen et al. "Quantum computational advantage with a programmable photonic processor". *Nature* 606.7912, pp. 75–81 (2022). ISSN: 1476-4687.

[190] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. "Trapped-ion quantum computing: Progress and challenges". *Appl. Phys. Rev.* 6.2, p. 021314 (2019).

[191] M Saffman. "Quantum computing with atomic qubits and Rydberg interactions: progress and challenges". *J. Phys. B: At. Mol. Opt. Phys.* 49.20, p. 202001 (2016).

[192] Loïc Henriet et al. "Quantum computing with neutral atoms". *Quantum* 4, p. 327 (2020). ISSN: 2521-327X.

[193] A Yu Kitaev. "Quantum computations: algorithms and error correction". *Russ. Math. Surv.* 52.6, pp. 1191–1249 (1997).

[194] Christopher M. Dawson and Michael A. Nielsen. *The Solovay-Kitaev algorithm*. 2005. arXiv: quant-ph/0505030 [quant-ph].

[195] Lorenzo Moro, Matteo G. A. Paris, Marcello Restelli, and Enrico Prati. "Quantum compiling by deep reinforcement learning". *Commun. Phys.* 4.1, p. 178 (2021). ISSN: 2399-3650.

[196] Borja Requena. *Compilation of quantum circuits*. PennyLane Demos (2023).

[197] W. K. Wootters and W. H. Zurek. "A single quantum cannot be cloned". *Nature* 299.5886, pp. 802–803 (1982). ISSN: 1476-4687.

[198] A. M. Steane. "Error Correcting Codes in Quantum Theory". *Phys. Rev. Lett.* 77 (5), pp. 793–797 (1996).

[199] Daniel Gottesman. *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation*. 2009. arXiv: 0904.2557 [quant-ph].

[200] A. D. Córcoles et al. "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits". *Nat. Commun.* 6.1, p. 6979 (2015).

[201] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. "Quantum error correction for the toric code using deep reinforcement learning". *Quantum* 3, p. 183 (2019). ISSN: 2521-327X.

[202] Ryan Sweke, Markus S Kesselring, Evert P L van Nieuwenburg, and Jens Eisert. "Reinforcement learning decoders for fault-tolerant quantum computation". *Mach. Learn.: Sci. Technol.* 2.2, p. 025005 (2021).

[203] V. V. Sivak et al. "Real-time quantum error correction beyond break-even". *Nature* 616.7955, pp. 50–55 (2023). ISSN: 1476-4687.

[204] Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt. "Reinforcement Learning with Neural Networks for Quantum Feedback". *Phys. Rev. X* 8 (3), p. 031084 (2018).

[205] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J. Briegel, and Nicolai Friis. "Optimizing Quantum Error Correction Codes with Reinforcement Learning". *Quantum* 3, p. 215 (2019). ISSN: 2521-327X.

[206] Agnes Valenti, Evert van Nieuwenburg, Sebastian Huber, and Eliska Greplova. "Hamiltonian learning for quantum error correction". *Phys. Rev. Res.* 1 (3), p. 033092 (2019).

[207] Murphy Yuezhen Niu, Sergio Boixo, Vadim N. Smelyanskiy, and Hartmut Neven. "Universal quantum control through deep reinforcement learning". *npj Quantum Inf.* 5.33, pp. 1–8 (2019). ISSN: 2056-6387.

[208] Yuval Baum et al. "Experimental Deep Reinforcement Learning for Error-Robust Gate-Set Design on a Superconducting Quantum Computer". *PRX Quantum* 2 (4), p. 040324 (2021).

[209] Pranav S. Mundada et al. "Experimental Benchmarking of an Automated Deterministic Error-Suppression Workflow for Quantum Algorithms". *Phys. Rev. Appl.* 20 (2), p. 024034 (2023).

[210] Jacob Biamonte et al. "Quantum machine learning". *Nature* 549.7671, pp. 195–202 (2017).

[211] Vedran Dunjko and Hans J Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". *Rep. Prog. Phys.* 81.7, p. 074001 (2018).

[212] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. "A rigorous and robust quantum speed-up in supervised machine learning". *Nat. Phys.* 17.9, pp. 1013–1017 (2021).

[213] Hsin-Yuan Huang et al. "Quantum advantage in learning from experiments". *Science* 376.6598, pp. 1182–1186 (2022).

[214] Junyu Liu et al. *Towards provably efficient quantum algorithms for large-scale machine-learning models*. 2023. arXiv: 2303.03428 [quant-ph].

[215] Maria Schuld and Nathan Killoran. "Quantum Machine Learning in Feature Hilbert Spaces". *Phys. Rev. Lett.* 122 (4), p. 040504 (2019).

[216] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces". *Nature* 567.7747, pp. 209–212 (2019).

[217] Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. "Parametrized Quantum Policies for Reinforcement Learning". *NeurIPS 2021 - Adv. Neural Inf. Process. Syst.* Pp. 28362–28375 (2021).

[218] Sofiene Jerbi, Lea M Trenkwalder, Hendrik Poulsen Nautrup, Hans J Briegel, and Vedran Dunjko. "Quantum enhancements for deep reinforcement learning in large spaces". *PRX Quantum* 2.1, p. 010328 (2021).

[219] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. "Data re-uploading for a universal quantum classifier". *Quantum* 4, p. 226 (2020). ISSN: 2521-327X.

[220] Maria Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. arXiv: 2101.11020 [quant-ph].

[221] Sofiene Jerbi et al. "Quantum machine learning beyond kernel methods". *Nat. Commun.* 14.1, p. 517 (2023).

[222] Joseph Bowles, Victoria J Wright, Máté Farkas, Nathan Killoran, and Maria Schuld. *Contextuality and inductive bias in quantum machine learning*. 2023. arXiv: 2302.01365 [quant-ph].

[223] Borja Requena, Gorka Muñoz-Gil, Maciej Lewenstein, Vedran Dunjko, and Jordi Tura. "Certificates of quantum many-body properties assisted by machine learning". *Phys. Rev. Res.* 5 (1), p. 013097 (2023).

[224] Borja Requena, Gorka Muñoz-Gil, and Jordi Tura. *BorjaRequena/BOUNCE*. Zenodo (2021).

[225] B. P. Lanyon et al. "Towards quantum chemistry on a quantum computer". *Nature Chem.* 2.2, pp. 106–111 (2010).

[226] P. J. J. O'Malley et al. "Scalable Quantum Simulation of Molecular Energies". *Phys. Rev. X* 6.3 (2016).

[227] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". *Nature* 549.7671, pp. 242–246 (2017).

[228] Steven R. White. "Density matrix formulation for quantum renormalization groups". *Physical Review Letters* 69.19, pp. 2863–2866 (1992).

[229] Steven R. White. "Density-matrix algorithms for quantum renormalization groups". *Phys. Rev. B* 48 (14), pp. 10345–10356 (1993).

[230] Carlos Bravo-Prieto, Josep Lumbreras-Zarapico, Luca Tagliacozzo, and José I. Latorre. "Scaling of variational quantum circuit depth for condensed matter systems". *Quantum* 4, p. 272 (2020). ISSN: 2521-327X.

[231] A. Garcia-Saez and J. I. Latorre. *Addressing hard classical problems with Adiabatically Assisted Variational Quantum Eigensolvers*. 2018. arXiv: 1806.02287 [quant-ph].

[232] Y. Herasymenko and T.E. O'Brien. "A diagrammatic approach to variational quantum ansatz construction". *Quantum* 5, p. 596 (2021). ISSN: 2521-327X.

[233] Jordi Tura. "Quantum Algorithms for Near-term Devices". *QANSWER 2020 QuANtum SoftWare Engineering & pRogramming*, pp. 38–50 (2020).

[234] Benjamin F. Schiffer, Jordi Tura, and J. Ignacio Cirac. "Adiabatic Spectroscopy and a Variational Quantum Adiabatic Algorithm". *PRX Quantum* 3 (2), p. 020347 (2022).

[235] Andrew C. Doherty, Pablo A. Parrilo, and Federico M. Spedalieri. "Complete family of separability criteria". *Phys. Rev. A* 69 (2), p. 022308 (2004).

[236] Miguel Navascués, Stefano Pironio, and Antonio Acín. "Bounding the Set of Quantum Correlations". *Phys. Rev. Lett.* 98 (1), p. 010401 (2007).

[237] F. Baccari, D. Cavalcanti, P. Wittek, and A. Acín. "Efficient Device-Independent Entanglement Detection for Multipartite Systems". *Phys. Rev. X* 7 (2), p. 021042 (2017).

[238] Asher Peres. "Separability Criterion for Density Matrices". *Phys. Rev. Lett.* 77 (8), pp. 1413–1415 (1996).

[239] Leonid Gurvits. "Classical Deterministic Complexity of Edmonds' Problem and Quantum Entanglement". *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 10–19 (2003).

[240] Michał Horodecki, Paweł Horodecki, and Ryszard Horodecki. "Separability of mixed states: necessary and sufficient conditions". *Physics Letters A* 223.1, pp. 1–8 (1996). ISSN: 0375-9601.

[241] Antonio Acín et al. "Device-Independent Security of Quantum Cryptography against Collective Attacks". *Phys. Rev. Lett.* 98 (23), p. 230501 (2007).

[242] Rodrigo Gallego et al. "Full randomness from arbitrarily deterministic events". *Nat. Commun.* 4.1, p. 2654 (2013). ISSN: 2041-1723.

[243] R. Augusiak, M. Demianowicz, M. Pawłowski, J. Tura, and A. Acín. "Elemental and tight monogamy relations in nonsignaling theories". *Phys. Rev. A* 90 (5), p. 052323 (2014).

[244] William Slofstra. "The set of quantum correlations is not closed". *Forum of Mathematics, Pi* 7, e1 (2019).

[245] Sandu Popescu and Daniel Rohrlich. "Quantum nonlocality as an axiom". *Foundations of Physics* 24.3, pp. 379–385 (1994). ISSN: 1572-9516.

[246] Gilles Brassard et al. "Limit on Nonlocality in Any World in Which Communication Complexity Is Not Trivial". *Phys. Rev. Lett.* 96 (25), p. 250401 (2006).

[247] Noah Linden, Sandu Popescu, Anthony J. Short, and Andreas Winter. "Quantum Nonlocality and Beyond: Limits from Nonlocal Computation". *Phys. Rev. Lett.* 99 (18), p. 180502 (2007).

[248] Marcin Pawłowski et al. "Information causality as a physical principle". *Nature* 461.7267, pp. 1101–1104 (2009). ISSN: 1476-4687.

[249] Miguel Navascués and Harald Wunderlich. "A glance beyond the quantum model". *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 466.2115, pp. 881–890 (2010). ISSN: 1364-5021.

[250] Rodrigo Gallego, Lars Erik Würflinger, Antonio Acín, and Miguel Navascués. "Quantum Correlations Require Multipartite Information Principles". *Phys. Rev. Lett.* 107 (21), p. 210403 (2011).

[251] T. Fritz et al. "Local orthogonality as a multipartite principle for quantum correlations". *Nature Communications* 4. Article, 2263 EP - (2013).

[252] Miguel Navascués, Yelena Guryanova, Matty J. Hoban, and Antonio Acín. "Almost quantum correlations". *Nat. Commun.* 6. Article, 6288 EP - (2015).

[253] Miguel Navascués, Stefano Pironio, and Antonio Acín. "A convergent hierarchy of semidefinite programs characterizing the set of quantum correlations". *New Journal of Physics* 10.7, p. 073013 (2008).

[254] S. Pironio, M. Navascués, and A. Acín. "Convergent Relaxations of Polynomial Optimization Problems with Noncommuting Variables". *SIAM Journal on Optimization* 20.5, pp. 2157–2180 (2010).

[255] P. W. Anderson. "Limits on the Energy of the Antiferromagnetic Ground State". *Physical Review* 83.6, pp. 1260–1260 (1951).

[256] L. Epele, H. Fanchiotti, C.A. García-Canal, and A.F. Pacheco. "Lower bound for the ground energy of spin systems". *Physica A: Statistical Mechanics and its Applications* 173.3, pp. 500–506 (1991). ISSN: 0378-4371.

[257] Gergely Gidofalvi and David A. Mazziotti. "Boson correlation energies via variational minimization with the two-particle reduced density matrix: ExactN-representability conditions for harmonic interactions". *Phys. Rev. A* 69.4 (2004).

[258] David A. Mazziotti. "Realization of Quantum Chemistry without Wave Functions through First-Order Semidefinite Programming". *Phys. Rev. Lett.* 93 (21), p. 213001 (2004).

[259] Tobias J. Osborne. *Lower Bound for the Ground-State Energy Density of a 1D Quantum Spin System*. 2006. arXiv: `cond-mat/0508428 [cond-mat.str-el]`.

[260] David A. Mazziotti. "Enhanced Constraints for Accurate Lower Bounds on Many-Electron Quantum Energies from Variational Two-Electron Reduced Density Matrix Theory". *Phys. Rev. Lett.* 117 (15), p. 153001 (2016).

[261] F. Uskov and O. Lychkovskiy. "A variational lower bound on the ground state of a many-body system and the squaring parametrization of density matrices". *Journal of Physics: Conference Series* 1163.1, p. 012057 (2019).

[262] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: A methodological tour d'horizon". *European Journal of Operational Research* 290.2, pp. 405–421 (2021). ISSN: 0377-2217.

[263] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer Networks". *Advances in Neural Information Processing Systems* (2015).

[264] Nikolaos Karalias and Andreas Loukas. "Erdos Goes Neural: an Unsupervised Learning Framework for Combinatorial Optimization on Graphs". *Advances in Neural Information Processing Systems*, pp. 6659–6672 (2020).

[265] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. "Reinforcement learning for combinatorial optimization: A survey". *Computers & Operations Research* 134, p. 105400 (2021). ISSN: 0305-0548.

[266] J. Eisert, M. Cramer, and M. B. Plenio. "Colloquium: Area laws for the entanglement entropy". *Rev. Mod. Phys.* 82 (1), pp. 277–306 (2010).

[267] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal. "What Limits the Simulation of Quantum Computers?" *Phys. Rev. X* 10 (4), p. 041038 (2020).

[268] Julia Kempe and Oded Regev. "3-local Hamitonian is QMA-complete". *Quantum Info. Comput.* 3.3, pp. 258–264 (2003). ISSN: 1533-7146.

[269] Julia Kempe, Alexei Kitaev, and Oded Regev. "The Complexity of the Local Hamiltonian Problem". *SIAM Journal on Computing* 35.5, pp. 1070–1097 (2006).

[270] Dorit Aharonov, Daniel Gottesman, Sandy Irani, and Julia Kempe. "The Power of Quantum Systems on a Line". *Communications in Mathematical Physics* 287.1, pp. 41–65 (2009).

[271] Norbert Schuch and Frank Verstraete. "Computational complexity of interacting electrons and fundamental limitations of density functional theory". *Nat. Phys.* 5.10, pp. 732–735 (2009).

[272] Jakub Czartowski, Konrad Szyma ński, Bartłomiej Gardas, Yan V. Fyodorov, and Karol Życzkowski. "Separability gap and large-deviation entanglement criterion". *Phys. Rev. A* 100 (4), p. 042326 (2019).

[273] Rolf Tarrach and Roser Valentí. "Exact lower bounds to the ground-state energy of spin systems: The two-dimensional S=1/2 antiferromagnetic Heisenberg model". *Phys. Rev. B* 41 (13), pp. 9611–9613 (1990).

[274] Anushya Chandran, Dagomir Kaszlikowski, Aditi Sen(De), Ujjwal Sen, and Vlatko Vedral. "Regional Versus Global Entanglement in Resonating-Valence-Bond States". *Phys. Rev. Lett.* 99 (17), p. 170502 (2007).

[275] Fabien Alet, Daniel Braun, and Grégoire Misguich. "Comment on "Regional Versus Global Entanglement in Resonating-Valence-Bond States"". *Phys. Rev. Lett.* 101 (24), p. 248901 (2008).

[276] Albert Aloy, Matteo Fadel, and Jordi Tura. "The quantum marginal problem for symmetric states: applications to variational optimization, nonlocality and self-testing". *New Journal of Physics* 23.3, p. 033026 (2021).

[277] Adam Sawicki, Michał Oszmaniec, and Marek Ku ś. "Critical sets of the total variance can detect all stochastic local operations and classical communication classes of multiparticle entanglement". *Phys. Rev. A* 86 (4), p. 040304 (2012).

[278] Michael Walter, Brent Doran, David Gross, and Matthias Christandl. "Entanglement Polytopes: Multiparticle Entanglement from Single-Particle Information". *Science* 340.6137, pp. 1205–1208 (2013).

[279] Adam Sawicki, Michał Oszmaniec, and Marek Kuś. "Convexity of momentum map, Morse index, and quantum entanglement". *Reviews in Mathematical Physics* 26.03, p. 1450004 (2014).

[280] Grigoriy Blekherman, Pablo A. Parrilo, and Rekha R. Thomas. *Semidefinite Optimization and Convex Algebraic Geometry*. Society for Industrial and Applied Mathematics (2012).

[281] Miguel Navascués, Masaki Owari, and Martin B. Plenio. "Power of symmetric extensions for entanglement detection". *Phys. Rev. A* 80 (5), p. 052306 (2009).

[282] Jos F. Sturm. "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones". *Optimization Methods and Software* 11.1-4, pp. 625–653 (1999).

[283] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. eng. Springer (1988).

[284] Farid Alizadeh. "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization". *SIAM Journal on Optimization* 5.1, pp. 13–51 (1995).

[285] S. Arora, E. Hazan, and S. Kale. "Fast algorithms for approximate semidefinite programming using the multiplicative weights update method". *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pp. 339–348 (2005).

[286] Fernando G.S.L. Brandao and Krysta M. Svore. "Quantum Speed-Ups for Solving Semidefinite Programs". *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 415–426 (2017).

[287] Tamás Kriváchy, Yu Cai, Joseph Bowles, Daniel Cavalcanti, and Nicolas Brunner. "High-speed batch processing of semidefinite programs with feed-forward neural networks". *New Journal of Physics* 23.10, p. 103034 (2021).

[288] Elliott Lieb, Theodore Schultz, and Daniel Mattis. "Two soluble models of an antiferromagnetic chain". *Annals of Physics* 16.3, pp. 407–466 (1961). ISSN: 0003-4916.

[289] Xiaoguang Wang. "Entanglement in the quantum Heisenberg XY model". *Phys. Rev. A* 64 (1), p. 012313 (2001).

[290] Xiaoguang Wang. "Thermal and ground-state entanglement in Heisenberg XX qubit rings". *Phys. Rev. A* 66 (3), p. 034302 (2002).

[291] A. De Pasquale et al. "XX model on the circle". *The European Physical Journal Special Topics* 160.1, pp. 127–138 (2008). ISSN: 1951-6401.

[292] P. Jordan and E. Wigner. "Über das Paulische Äquivalenzverbot". *Zeitschrift für Physik* 47.9-10, pp. 631–651 (1928).

[293] Michael A Nielsen. "The Fermionic canonical commutation relations and the Jordan-Wigner transform". *School of Physical Sciences The University of Queensland* 59 (2005).

[294] Mari-Carmen Bañuls, J. Ignacio Cirac, and Michael M. Wolf. "Entanglement in fermionic systems". *Phys. Rev. A* 76 (2), p. 022311 (2007).

[295] J. Tura et al. "Energy as a Detector of Nonlocality of Many-Body Spin Systems". *Phys. Rev. X* 7 (2), p. 021005 (2017).

[296] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press (2022).

[297] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". *Science* 220.4598, pp. 671–680 (1983).

[298] Matthew E. Taylor and Peter Stone. "Transfer Learning for Reinforcement Learning Domains: A Survey". *Journal of Machine Learning Research* 10.56, pp. 1633–1685 (2009).

[299] Otfried Gühne and Géza Tóth. "Entanglement detection". *Physics Reports* 474.1, pp. 1–75 (2009). ISSN: 0370-1573.

[300] G. Tóth and O. Gühne. "Detection of multipartite entanglement with two-body correlations". *Applied Physics B* 82.2, pp. 237–241 (2006). ISSN: 1432-0649.

[301] Otfried Gühne, Géza Tóth, and Hans J Briegel. "Multipartite entanglement in spin chains". *New Journal of Physics* 7.1, p. 229 (2005).

[302] Nengkun Yu. "Separability of a mixture of Dicke states". *Phys. Rev. A* 94 (6), p. 060101 (2016).

[303] Jordi Tura, Albert Aloy, Ruben Quesada, Maciej Lewenstein, and Anna Sanpera. "Separability of diagonal symmetric states: a quadratic conic optimization problem". *Quantum* 2, p. 45 (2018). ISSN: 2521-327X.

[304] Carlo Marconi, Albert Aloy, Jordi Tura, and Anna Sanpera. "Entangled symmetric states and copositive matrices". *Quantum* 5, p. 561 (2021). ISSN: 2521-327X.

[305] A. Aloy et al. "Device-Independent Witnesses of Entanglement Depth from Two-Body Correlators". *Phys. Rev. Lett.* 123 (10), p. 100507 (2019).

[306] J. Tura et al. "Optimization of device-independent witnesses of entanglement depth from two-body correlators". *Phys. Rev. A* 100 (3), p. 032307 (2019).

[307] Géza Tóth. "Entanglement witnesses in spin models". *Phys. Rev. A* 71 (1), p. 010301 (2005).

[308] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. GitHub (2018).

[309] Borja Requena. *Gate calibration with reinforcement learning*. PennyLane Demos (2024).

[310] Jens Koch et al. "Charge-insensitive qubit design derived from the Cooper pair box". *Phys. Rev. A* 76 (4), p. 042319 (2007).

[311] P. Krantz et al. "A quantum engineer's guide to superconducting qubits". *Applied Physics Reviews* 6.2, p. 021318 (2019). ISSN: 1931-9401.

[312] Chad Rigetti and Michel Devoret. "Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies". *Phys. Rev. B* 81 (13), p. 134507 (2010).

[313] Jerry M. Chow et al. "Simple All-Microwave Entangling Gate for Fixed-Frequency Superconducting Qubits". *Phys. Rev. Lett.* 107 (8), p. 080502 (2011).

[314] Sarah Sheldon, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. "Procedure for systematically tuning up cross-talk in the cross-resonance gate". *Phys. Rev. A* 93 (6), p. 060302 (2016).

[315] Andre R. R. Carvalho, Harrison Ball, Michael J. Biercuk, Michael R. Hush, and Felix Thomsen. "Error-Robust Quantum Logic Optimization Using a Cloud Quantum Computer Interface". *Phys. Rev. Appl.* 15 (6), p. 064054 (2021).

[316] Easwar Magesan and Jay M. Gambetta. "Effective Hamiltonian models of the cross-resonance gate". *Phys. Rev. A* 101 (5), p. 052308 (2020).

[317] M. A. Rol et al. "Time-domain characterization and correction of on-chip distortion of control pulses in a quantum processor". *Applied Physics Letters* 116.5, p. 054001 (2020). ISSN: 0003-6951.

[318] Matthew Reagor et al. "Demonstration of universal parametric entangling gates on a multi-qubit lattice". *Science Advances* 4.2, eaao3603 (2018).

[319] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. "Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits". *Phys. Rev. Lett.* 103 (11), p. 110501 (2009).

[320] Karl Pearson. "The Problem of the Random Walk". *Nature* 72.1865, pp. 294–294 (1905). ISSN: 1476-4687.

[321] Farina Kindermann et al. "Nonergodic diffusion of single atoms in a periodic potential". *Nat. Phys.* 13.2, pp. 137–141 (2017). ISSN: 1745-2481.

[322] Felix Höfling and Thomas Franosch. "Anomalous transport in the crowded world of biological cells". *Reports on Progress in Physics* 76.4, p. 046602 (2013).

[323] Ido Golding and Edward C. Cox. "Physical Nature of Bacterial Cytoplasm". *Phys. Rev. Lett.* 96 (9), p. 098102 (2006).

[324] Carlo Manzo et al. "Weak Ergodicity Breaking of Receptor Motion in Living Cells Stemming from Random Diffusivity". *Phys. Rev. X* 5 (1), p. 011021 (2015).

[325] Nicolas E. Humphries, Henri Weimerskirch, Nuno Queiroz, Emily J. Southall, and David W. Sims. "Foraging success of biological Lévy flights recorded in situ". *Proceedings of the National Academy of Sciences* 109.19, pp. 7169–7174 (2012).

[326] Vasiliki Plerou, Parameswaran Gopikrishnan, Luís A. Nunes Amaral, Xavier Gabaix, and H. Eugene Stanley. "Economic fluctuations and anomalous diffusion". *Phys. Rev. E* 62 (3), R3023–R3026 (2000).

[327] Oliver Lüdtke, Brent W Roberts, Ulrich Trautwein, and Gabriel Nagy. "A random walk down university avenue: life paths, life events, and personality trait change at the transition to university life." *Journal of Personality and Social Psychology* 101.3, pp. 620–637 (2011).

[328] Robert Brown. "XXVII. A brief account of microscopical observations made in the months of June, July and August 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies". *The Philosophical Magazine* 4.21, pp. 161–173 (1828).

[329] Albert Einstein. "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen". *Annalen der Physik* 8, pp. 549–560 (1905).

[330] Albert Einstein. "Zur theorie der brownschen bewegung". *Annalen der Physik* 2, pp. 371–381 (1906).

[331] Marian Von Smoluchowski. "Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen". *Annalen der physik* 326.14, pp. 756–780 (1906).

[332] Richard Von Mises. "Fundamentalsätze der Wahrscheinlichkeitsrechnung". *Math. Z.* 4, pp. 1–97 (1919).

[333] Elliott W Montroll and George H Weiss. "Random walks on lattices. II". *Journal of Mathematical Physics* 6.2, pp. 167–181 (1965).

[334] Ralf Metzler, Jae-Hyung Jeon, Andrey G Cherstvy, and Eli Barkai. "Anomalous diffusion models and their properties: non-stationarity, non-ergodicity, and ageing at the centenary of single particle tracking". *Phys. Chem. Chem. Phys.* 16.44, pp. 24128–24164 (2014).

[335] Ludwig Boltzmann. *Vorlesungen über Gastheorie: Th. Theorie van der Waals'; Gase mit zusammengesetzten Molekülen; Gasdissociation; Schlussbemerkungen*. JA Barth (1898).

[336] J. Bouchaud. "Weak ergodicity breaking and aging in disordered systems". *Journal de Physique I* 2.9, pp. 1705–1713 (1992).

[337] Harvey Scher and Elliott W. Montroll. "Anomalous transit-time dispersion in amorphous solids". *Phys. Rev. B* 12 (6), pp. 2455–2477 (1975).

[338] Ryszard Kutner and Jaume Masoliver. "The continuous time random walk, still trendy: fifty-year history, state of art and outlook". *The European Physical Journal B* 90.3, p. 50 (2017). ISSN: 1434-6036.

[339] J. Klafter and G. Zumofen. "Lévy statistics in a Hamiltonian system". *Phys. Rev. E* 49 (6), pp. 4873–4877 (1994).

[340] V. Zaburdaev, S. Denisov, and J. Klafter. "Lévy walks". *Rev. Mod. Phys.* 87 (2), pp. 483–530 (2015).

[341] P. Massignan et al. "Nonergodic Subdiffusion from Brownian Motion in an Inhomogeneous Medium". *Phys. Rev. Lett.* 112 (15), p. 150603 (2014).

[342] Andrei Nikolaevitch Kolmogorov. "Curves in Hilbert space which are invariant with respect to a one-parameter group of motions". *Dokl. Akad. Nauk SSSR*, pp. 6–9 (1940).

[343] Benoit B. Mandelbrot and John W. Van Ness. "Fractional Brownian Motions, Fractional Noises and Applications". *SIAM Review* 10.4, pp. 422–437 (1968).

[344] Weichun Pan et al. "Viscoelasticity in Homogeneous Protein Solutions". *Phys. Rev. Lett.* 102 (5), p. 058101 (2009).

[345] Julia F. Reverey et al. "Superdiffusion dominates intracellular particle motion in the supercrowded cytoplasm of pathogenic Acanthamoeba castellanii". *Scientific Reports* 5.1, p. 11690 (2015). ISSN: 2045-2322.

[346] Paul Langevin. "Sur la théorie du mouvement brownien". *CR Acad Sci (Paris)* 146, p. 530 (1908).

[347] S. C. Lim and S. V. Muniandy. "Self-similar Gaussian processes for modeling anomalous diffusion". *Phys. Rev. E* 66 (2), p. 021114 (2002).

[348] Jae-Hyung Jeon, Aleksei V. Chechkin, and Ralf Metzler. "Scaled Brownian motion: a paradoxical process with a time dependent diffusivity for the description of anomalous diffusion". *Phys. Chem. Chem. Phys.* 16 (30), pp. 15811–15817 (2014).

[349] Anna Bodrova, Aleksei V. Chechkin, Andrey G. Cherstvy, and Ralf Metzler. "Quantifying non-ergodic dynamics of force-free granular gases". *Phys. Chem. Chem. Phys.* 17 (34), pp. 21791–21798 (2015).

[350] Anna S. Bodrova et al. "Underdamped scaled Brownian motion: (non-)existence of the overdamped limit in anomalous diffusion". *Scientific Reports* 6.1, p. 30520 (2016). ISSN: 2045-2322.

[351] Jae-Hyung Jeon, Matti Javanainen, Hector Martinez-Seara, Ralf Metzler, and Ilpo Vattulainen. "Protein Crowding in Lipid Bilayers Gives Rise to Non-Gaussian Anomalous Lateral Diffusion of Phospholipids and Proteins". *Phys. Rev. X* 6 (2), p. 021006 (2016).

[352] Carlo Manzo and Maria F Garcia-Parajo. "A review of progress in single particle tracking: from methods to biophysical insights". *Reports on Progress in Physics* 78.12, p. 124601 (2015).

[353] Xavier Michalet and Andrew J. Berglund. "Optimal diffusion coefficient estimation in single-particle tracking". *Phys. Rev. E* 85 (6), p. 061916 (2012).

[354] Christian L. Vestergaard, Paul C. Blainey, and Henrik Flyvbjerg. "Optimal estimation of diffusion coefficients from single-particle trajectories". *Phys. Rev. E* 89 (2), p. 022726 (2014).

[355] Douglas S Martin, Martin B Forstner, and Josef A Käs. "Apparent subdiffusion inherent to single particle tracking". *Biophysical journal* 83.4, pp. 2109–2117 (2002).

[356] Eldad Kepten, Irena Bronshtein, and Yuval Garini. "Improved estimation of anomalous diffusion exponents in single-particle tracking experiments". *Phys. Rev. E* 87 (5), p. 052713 (2013).

[357] Eldad Kepten, Aleksander Weron, Grzegorz Sikora, Krzysztof Burnecki, and Yuval Garini. "Guidelines for the Fitting of Anomalous Diffusion Mean Square Displacement Graphs from Single Particle Tracking Experiments". *PLOS ONE* 10.2, pp. 1–10 (2015).

[358] Nicolas Chenouard et al. "Objective comparison of particle tracking methods". *Nat. Methods* 11.3, pp. 281–289 (2014). ISSN: 1548-7105.

[359] Diego Krapf et al. "Spectral Content of a Single Non-Brownian Trajectory". *Phys. Rev. X* 9 (1), p. 011019 (2019).

[360] Vittoria Sposini et al. "Towards a robust criterion of anomalous diffusion". *Commun. Phys.* 5.1, p. 305 (2022). ISSN: 2399-3650.

[361] Samudrajit Thapa, Michael A. Lomholt, Jens Krog, Andrey G. Cherstvy, and Ralf Metzler. "Bayesian analysis of single-particle tracking data using the nested-sampling algorithm: maximum-likelihood model selection applied to stochastic-diffusivity data". *Phys. Chem. Chem. Phys.* 20 (46), pp. 29018–29037 (2018).

[362] Henrik Seckler, Janusz Szwabiński, and Ralf Metzler. "Machine-Learning Solutions for the Analysis of Single-Particle Diffusion Trajectories". *The Journal of Physical Chemistry Letters* 14.35. PMID: 37646323, pp. 7910–7923 (2023).

[363] Naor Granik et al. "Single-Particle Diffusion Characterization by Deep Learning". *Biophysical Journal* 117.2, pp. 185–192 (2019). ISSN: 0006-3495.

[364] Stefano Bo, Falko Schmidt, Ralf Eichhorn, and Giovanni Volpe. "Measurement of anomalous diffusion using recurrent neural networks". *Phys. Rev. E* 100 (1), p. 010102 (2019).

[365] Patrycja Kowalek, Hanna Loch-Olszewska, and Janusz Szwabi ński. "Classification of diffusion modes in single-particle tracking data: Feature-based versus deep-learning approach". *Phys. Rev. E* 100 (3), p. 032410 (2019).

[366] Gorka Muñoz-Gil, Miguel Angel Garcia-March, Carlo Manzo, José D Martín-Guerrero, and Maciej Lewenstein. "Single trajectory characterization via machine learning". *New J. Phys.* 22.1, p. 013010 (2020).

[367] Gorka Muñoz-Gil et al. "Objective comparison of methods to decode anomalous diffusion". *Nat. Commun.* 12.1, p. 6253 (2021). ISSN: 2041-1723.

[368] Alessia Gentili and Giorgio Volpe. "Characterization of anomalous diffusion classical statistics powered by deep learning (CONDOR)". *J. Phys. A: Math. Theor.* 54.31, p. 314003 (2021).

[369] Aykut Argun, Giovanni Volpe, and Stefano Bo. "Classification, inference and segmentation of anomalous diffusion with recurrent neural networks". *J. Phys. A: Math. Theor.* 54.29, p. 294003 (2021).

[370] Òscar Garibo-i-Orts, Alba Baeza-Bosca, Miguel A Garcia-March, and J Alberto Conejero. "Efficient recurrent neural network methods for anomalously diffusing single particle short and noisy trajectories". *J. Phys. A: Math. Theor.* 54.50, p. 504002 (2021).

[371] Dezhong Li, Qiujin Yao, and Zihan Huang. "WaveNet-based deep neural networks for the characterization of anomalous diffusion (WADNet)". *J. Phys. A: Math. Theor.* 54.40, p. 404003 (2021).

[372] Hippolyte Verdier et al. "Learning physical properties of anomalous random walks using graph neural networks". *Journal of Physics A: Mathematical and Theoretical* 54.23, p. 234001 (2021).

[373] Vida Jamali et al. "Anomalous nanoparticle surface diffusion in LCTEM is revealed by deep learning-assisted analysis". *Proceedings of the National Academy of Sciences* 118.10, e2017616118 (2021).

[374] Gorka Muñoz-Gil et al. "Stochastic particle unbinding modulates growth dynamics and size of transcription factor condensates in living cells". *Proceedings of the National Academy of Sciences* 119.31, e2200667119 (2022).

[375] Henrik Seckler and Ralf Metzler. "Bayesian deep learning for error estimation in the analysis of anomalous diffusion". *Nat. Commun.* 13.1, p. 6717 (2022).

[376] Gabriel Fernández-Fernández, Carlo Manzo, Maciej Lewenstein, Alexandre Dauphin, and Gorka Muñoz-Gil. *Learning minimal representations of stochastic processes with variational autoencoders*. 2023. arXiv: 2307.11608 [cond-mat.soft].

[377] Hélène Kabbech and Ihor Smal. "Identification of Diffusive States in Tracking Applications Using Unsupervised Deep Learning Methods". *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pp. 1–4 (2022).

[378] Jesús Pineda et al. "Geometric deep learning reveals the spatiotemporal features of microscopic motion". *Nat. Mach. Intell.* 5.1, pp. 71–82 (2023). ISSN: 2522-5839.

[379] Benjamin Midtvedt et al. "Quantitative digital microscopy with deep learning". *Appl. Phys. Rev.* 8.1, p. 011310 (2021). ISSN: 1931-9401.

[380] Borja Requena. *BorjaRequena/AnDi-unicorns*. GitHub (2020).

[381] Gorka Muñoz-Gil, Borja Requena, Giovanni Volpe, Miguel Angel Garcia-March, and Carlo Manzo. *ANDI_datasets Python library (github.com/AnDiChallenge/andi_datasets)*. Zenodo (2021).

[382] Gorka Muñoz et al. *Quantitative evaluation of methods to analyze motion changes in single-particle experiments*. 2023. arXiv: 2311.18100 [cond-mat.soft].

[383] Philipp G Meyer, Erez Aghion, and Holger Kantz. "Decomposing the effect of anomalous diffusion enables direct calculation of the Hurst exponent and model classification for single random paths". *Journal of Physics A: Mathematical and Theoretical* 55.27, p. 274001 (2022).

[384] Seongyu Park, Samudrajit Thapa, Yeongjin Kim, Michael A Lomholt, and Jae-Hyung Jeon. "Bayesian inference of Lévy walks via hidden Markov models". *Journal of Physics A: Mathematical and Theoretical* 54.48, p. 484001 (2021).

[385] Samudrajit Thapa et al. "Bayesian inference of scaled versus fractional Brownian motion". *Journal of Physics A: Mathematical and Theoretical* 55.19, p. 194003 (2022).

[386] Carlo Manzo. "Extreme learning machine for the characterization of anomalous diffusion from single trajectories (AnDi-ELM)". *Journal of Physics A: Mathematical and Theoretical* 54.33, p. 334002 (2021).

[387] Patrycja Kowalek, Hanna Loch-Olszewska, Łukasz Łaszczuk, Jarosław Opała, and Janusz Szwabiński. "Boosting the performance of anomalous diffusion classifiers with the proper choice of features". *Journal of Physics A: Mathematical and Theoretical* 55.24, p. 244005 (2022).

[388] Nicolas Firbas, Òscar Garibo-i-Orts, Miguel Ángel Garcia-March, and J Alberto Conejero. "Characterization of anomalous diffusion through convolutional transformers". *Journal of Physics A: Mathematical and Theoretical* 56.1, p. 014001 (2023).

[389] Borja Requena et al. "Inferring pointwise diffusion properties of single trajectories with deep learning". *Biophysical Journal* 122.22, pp. 4360–4369 (2023). ISSN: 0006-3495.

[390] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems* (2019).

[391] Jeremy Howard and Sylvain Gugger. "Fastai: A Layered API for Deep Learning". *Information* 11.2 (2020). ISSN: 2078-2489.

[392] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[393] Leslie N. Smith and Nicholay Topin. "Super-convergence: very fast training of neural networks using large learning rates". *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, pp. 369–386 (2019).

[394] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *ICLR (Poster)* (2015).

[395] Borja Requena and Gorka Muñoz-Gil. *BorjaRequena/step*. Zenodo (2022).

[396] Leonhard Möckl, Don C. Lamb, and Christoph Bräuchle. "Super-resolved Fluorescence Microscopy: Nobel Prize in Chemistry 2014 for Eric Betzig, Stefan Hell, and William E. Moerner". *Angewandte Chemie International Edition* 53.51, pp. 13972–13977 (2014).

[397] Shuhui Yin, Nancy Song, and Haw Yang. "Detection of Velocity and Diffusion Coefficient Change Points in Single-Particle Trajectories". *Biophysical Journal* 115.2, pp. 217–229 (2018). ISSN: 0006-3495.

[398] Suvrajit Saha et al. "Diffusion of GPI-anchored proteins is influenced by the activity of dynamic cortical actin". *Molecular Biology of the Cell* 26.22. PMID: 26378258, pp. 4033–4045 (2015).

[399] Nirmalya Bag, Shuangru Huang, and Thorsten Wohland. "Plasma Membrane Organization of Epidermal Growth Factor Receptor in Resting and Ligand-Bound States". *Biophysical Journal* 109.9, pp. 1925–1936 (2015). ISSN: 0006-3495.

[400] Shalini T. Low-Nam et al. "ErbB1 dimerization is promoted by domain co-confinement and stabilized by ligand binding". *Nat. Struct. Mol. Biol.* 18.11, pp. 1244–1249 (2011). ISSN: 1545-9985.

[401] Adal Sabri, Xinran Xu, Diego Krapf, and Matthias Weiss. "Elucidating the Origin of Heterogeneous Anomalous Diffusion in the Cytoplasm of Mammalian Cells". *Phys. Rev. Lett.* 125 (5), p. 058101 (2020).

[402] Thomas J. Lampo, Stella Stylianidou, Mikael P. Backlund, Paul A. Wiggins, and Andrew J. Spakowitz. "Cytoplasmic RNA-Protein Particles Exhibit Non-Gaussian Subdiffusive Behavior". *Biophysical Journal* 112.3, pp. 532–542 (2017). ISSN: 0006-3495.

[403] Anthony R. Vega, Spencer A. Freeman, Sergio Grinstein, and Khuloud Jaqaman. "Multistep Track Segmentation and Motion Classification for Transient Mobility Analysis". *Biophysical Journal* 114.5, pp. 1018–1025 (2018). ISSN: 0006-3495.

[404] Yann Lanoiselée and Denis S. Grebenkov. "Unraveling intermittent features in single-particle trajectories by a local convex hull method". *Phys. Rev. E* 96 (2), p. 022144 (2017).

[405] Jonathan E. Bronson, Jingyi Fei, Jake M. Hofman, Ruben L. Gonzalez Jr., and Chris H. Wiggins. "Learning Rates and States from Biophysical Time Series: A Bayesian Approach to Model Selection and Single-Molecule FRET Data". *Biophysical Journal* 97.12, pp. 3196–3205 (2009). ISSN: 0006-3495.

[406] Fredrik Persson, Martin Lindén, Cecilia Unoson, and Johan Elf. "Extracting intracellular diffusive states and transition rates from single-molecule tracking data". *Nat. Methods* 10.3, pp. 265–269 (2013). ISSN: 1548-7105.

[407] Nilah Monnier et al. "Inferring transient particle transport dynamics in live cells". *Nat. Methods* 12.9, pp. 838–840 (2015). ISSN: 1548-7105.

[408] Hippolyte Verdier, Fran çois Laurent, Alhassan Cassé, Christian L. Vestergaard, and Jean-Baptiste Masson. "Variational inference of fractional Brownian motion with linear computational complexity". *Phys. Rev. E* 106 (5), p. 055311 (2022).

[409] Marloes Arts, Ihor Smal, Maarten W. Paul, Claire Wyman, and Erik Meijering. "Particle Mobility Analysis Using Deep Learning and the Moment Scaling Spectrum". *Sci. Rep.* 9.1, p. 17160 (2019). ISSN: 2045-2322.

[410] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". *Advances in Neural Information Processing Systems* (2014).

[411] Scott Reed et al. "A Generalist Agent". *Transactions on Machine Learning Research*. Featured Certification, Outstanding Certification (2022). ISSN: 2835-8856.

[412] Ross Taylor et al. *Galactica: A Large Language Model for Science*. 2022. arXiv: 2211.09085 [cs.CL].

[413] Héctor Buena Maizón and Francisco J Barrantes. "A deep learning-based approach to model anomalous diffusion of membrane proteins: the case of the nicotinic acetylcholine receptor". *Briefings in Bioinformatics* 23.1 (2021). ISSN: 1477-4054.

[414] Alain Celisse, Guillemette Marot, Morgane Pierre-Jean, and GJ Rigaill. "New efficient algorithms for multiple change-point detection with reproducing kernels". *Computational Statistics and Data Analysis* 128, pp. 200–220 (2018).

[415] Sylvain Arlot, Alain Celisse, and Zaid Harchaoui. "A Kernel Multiple Change-point Algorithm via Model Selection". *Journal of Machine Learning Research* 20.162, pp. 1–56 (2019).

[416] Charles Truong, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods". *Signal Processing* 167, p. 107299 (2020). ISSN: 0165-1684.

[417] Carlo Manzo et al. "The Neck Region of the C-type Lectin DC-SIGN Regulates Its Surface Spatiotemporal Organization and Virus-binding Capacity on Antigen-presenting Cells". *Journal of Biological Chemistry* 287.46, pp. 38946–38955 (2012). ISSN: 0021-9258.

[418] D. Montiel, H. Cang, and H. Yang. "Quantitative Characterization of Changes in Dynamical Behavior for Single-Particle Tracking Studies". *The Journal of Physical Chemistry B* 110.40. PMID: 17020359, pp. 19763–19770 (2006).

[419] Taka A. Tsunoyama et al. "Super-long single-molecule tracking reveals dynamic-anchorage-induced integrin function". *Nat. Chem. Biol.* 14.5, pp. 497–506 (2018). ISSN: 1552-4469.

[420] Pakorn Kanchanawong and David A. Calderwood. "Organization, dynamics and mechanoregulation of integrin-mediated cell–ECM adhesions". *Nat. Rev. Mol. Cell Biol.* 24.2, pp. 142–161 (2023). ISSN: 1471-0080.

[421] Olivier Rossier et al. "Integrins β1 and β3 exhibit distinct dynamic nanoscale organizations inside focal adhesions". *Nat. Cell Biol.* 14.10, pp. 1057–1067 (2012). ISSN: 1476-4679.

[422] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171 (2011).

[423] Borja Requena, Giovanni Cassani, Jacopo Tagliabue, Ciro Greco, and Lucas Lacasa. "Shopper intent prediction from clickstream e-commerce data with minimal browsing information". *Sci. Rep.* 10.1, p. 16983 (2020). ISSN: 2045-2322.

[424] Borja Requena, Giovanni Cassani, Jacopo Tagliabue, Ciro Greco, and Lucas Lacasa. *coveooss/shopper-intent-prediction-nature-2020*. GitHub (2020).

[425] Zhenzhou Wu, Bao Hong Tan, Rubing Duan, Yong Liu, and Rick Siow Mong Goh. "Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns". *Proceedings of the 2015 International ACM Recommender Systems Challenge* (2015).

[426] H. Brendan McMahan et al. "Ad Click Prediction: A View from the Trenches". *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1222–1230 (2013).

[427] Dimitris J. Bertsimas, Adam J. Mersereau, and Nitin R. Patel. "Dynamic Classification of Online Customers". *Proceedings of the 2003 SIAM International Conference on Data Mining (SDM)*, pp. 107–118 (2003).

[428] Arthur Toth, Louis Tan, Giuseppe Di Fabbrizio, and Ankur Datta. "Predicting Shopping Behavior with Mixture of RNNs". *Proceedings of the SIGIR 2017 Workshop on eCommerce (ECOM 17)* (2017).

[429] Aditya Awalkar, Ibrahim Ahmed, and Tejas Nevrekar. "Prediction of user's purchases using clickstream data". *International Journal of Engineering Science and Computing* 6.4, pp. 4044–4046 (2016).

[430] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. "A Brief Survey on Sequence Classification". *SIGKDD Explor. Newsl.* 12.1, pp. 40–48 (2010). ISSN: 1931-0145.

[431] Jannick Dyrløv Bendtsen, Lars Juhl Jensen, Nikolaj Blom, Gunnar von Heijne, and Søren Brunak. "Feature-based prediction of non-classical and leaderless protein secretion". *Protein Engineering, Design and Selection* 17.4, pp. 349–356 (2004). ISSN: 1741-0126.

[432] F Lotte et al. "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update". *Journal of Neural Engineering* 15.3, p. 031005 (2018).

[433] I. Amed et al. *The State of Fashion 2019: A year of awakening*. 2019.

[434] *E-commerce share of total retail revenue in the United States as of February 2019, by product category*. Accessed: 2019-04-22. 2019.

[435] Jacopo Tagliabue, Bingqing Yu, and Marie Beaulieu. "How to Grow a (Product) Tree: Personalized Category Suggestions for eCommerce Type-Ahead". *Proceedings of the 3rd Workshop on e-Commerce and NLP*, pp. 7–18 (2020).

[436] Jacopo Iacovacci and Lucas Lacasa. "Sequential visibility-graph motifs". *Phys. Rev. E* 93 (4), p. 042309 (2016).

[437] Lucas Lacasa, Bartolo Luque, Fernando Ballesteros, Jordi Luque, and Juan Carlos Nuño. "From time series to complex networks: The visibility graph". *Proceedings of the National Academy of Sciences* 105.13, pp. 4972–4975 (2008).

[438] L. Lacasa, B. Luque, J. Luque, and J. C. Nuño. "The visibility graph: A new method for estimating the Hurst exponent of fractional Brownian motion". *Europhysics Letters* 86.3, p. 30001 (2009).

[439] Urie Bronfenbrenner. "Toward an experimental ecology of human development." *American Psychologist* 32.7, pp. 513–531 (1977).

[440] Pawel Ogonowski. *2 Ecommerce Conversion Rate Statistics (Updated January 2021)*. 2021.

[441] Guozhu Dong and Jian Pei. *Sequence data mining*. Springer Science (2007). ISSN: 1386-2944.

[442] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque. "Horizontal visibility graphs: Exact results for random time series". *Phys. Rev. E* 80 (4), p. 046103 (2009).

[443] M. E. J. Newman. "The Structure and Function of Complex Networks". *SIAM Review* 45.2, pp. 167–256 (2003).

[444] Gregory Gutin, Toufik Mansour, and Simone Severini. "A characterization of horizontal visibility graphs and combinatorics on words". *Physica A: Statistical Mechanics and its Applications* 390.12, pp. 2421–2428 (2011). ISSN: 0378-4371.

[445] Bartolo Luque and Lucas Lacasa. "Canonical horizontal visibility graphs are uniquely determined by their degree sequence". *The European Physical Journal Special Topics* 226.3, pp. 383–389 (2017). ISSN: 1951-6401.

[446] Lucas Lacasa. "On the degree distribution of horizontal visibility graphs associated with Markov processes and dynamical systems: diagrammatic and variational approaches". *Nonlinearity* 27.9, p. 2063 (2014).

[447] Lucas Lacasa and Wolfram Just. "Visibility graphs and symbolic dynamics". *Physica D: Nonlinear Phenomena* 374-375, pp. 35–44 (2018). ISSN: 0167-2789.

[448] Bartolo Luque, Lucas Lacasa, Fernando J. Ballesteros, and Alberto Robledo. "Analytical properties of horizontal visibility graphs in the Feigenbaum scenario". *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1, p. 013109 (2012). ISSN: 1054-1500.

[449] Ángel M. Núñez, Bartolo Luque, Lucas Lacasa, Jose Patricio Gómez, and Alberto Robledo. "Horizontal visibility graphs generated by type-I intermittency". *Phys. Rev. E* 87 (5), p. 052801 (2013).

[450] B. Luque, F. J. Ballesteros, A. M. Núñez, and A. Robledo. "Quasiperiodic Graphs: Structural Design, Scaling and Entropic Properties". *J. of Nonlinear Sci.* 23.2, pp. 335–342 (2013). ISSN: 1432-1467.

[451] Mehran Ahmadlou, Hojjat Adeli, and Anahita Adeli. "New diagnostic EEG markers of the Alzheimer's disease using visibility graph". *J. Neural Transm.* 117.9, pp. 1099–1109 (2010). ISSN: 1435-1463.

[452] Speranza Sannino, Sebastiano Stramaglia, Lucas Lacasa, and Daniele Marinazzo. "Visibility graphs for fMRI data: Multiplex temporal graphs and their modulations across resting-state networks". *Network Neuroscience* 1.3, pp. 208–221 (2017). ISSN: 2472-1751.

[453] Meenatchidevi Murugesan and R. I. Sujith. "Combustion noise is scale-free: transition from scale-free to order at the onset of thermoacoustic instability". *Journal of Fluid Mechanics* 772, pp. 225–245 (2015).

[454] Pouya Manshour, M Reza Rahimi Tabar, and Joachim Peinke. "Fully developed turbulence in the view of horizontal visibility graphs". *J. Stat. Mech.* 2015.8, P08031 (2015).

[455] Y. Zou, R. V. Donner, N. Marwan, M. Small, and J. Kurths. "Long-term changes in the north–south asymmetry of solar activity: a nonlinear dynamics characterization using visibility graphs". *Nonlinear Processes in Geophysics* 21.6, pp. 1113–1126 (2014).

[456] Jacopo Iacovacci and Lucas Lacasa. "Visibility Graphs for Image Processing". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4, pp. 974–987 (2020).

[457] Jacopo Iacovacci and Lucas Lacasa. "Sequential motif profile of natural visibility graphs". *Phys. Rev. E* 94 (5), p. 052309 (2016).

[458] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794 (2016).

[459] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". *Advances in Neural Information Processing Systems* (2017).

[460] Scott M. Lundberg et al. "From local explanations to global understanding with explainable AI for trees". *Nat. Mach. Intell.* 2.1, pp. 56–67 (2020). ISSN: 2522-5839.

[461] Alexey Chervonenkis. "Early History of Support Vector Machines". In: Springer Berlin, Heidelberg, 2013, pp. 13–20. ISBN: 978-3-642-41135-9.

[462] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". *Chemometrics and Intelligent Laboratory Systems* 2.1. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists, pp. 37–52 (1987). ISSN: 0169-7439.

[463] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. "UMAP: Uniform Manifold Approximation and Projection". *Journal of Open Source Software* 3.29, p. 861 (2018).

[464] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].