University of Pisa

Department of Mathematics

Master Degree in Mathematics

Reinforcement Learning for Conformal Field Theories

Master Thesis

Supervisors
Prof. Davide Bacciu
Dr. Andrea Cossu
Dr. Pietro Ferrero

Candidate
Alessandro Trenta

Academic year 2022–2023

Reinforcement Learning for Conformal Field Theories

Candidate:
Alessandro Trenta
Supervisors:
Prof. Davide Bacciu
Dr. Andrea Cossu
Dr. Pietro Ferrero

Table of Contents

Chapter 1

Introduction

Conformal Field Theories (CFTs) are particular kinds of Quantum Field Theories (QFTs) that, in addition to relativistic invariance under the Poincaré group of transformations, require invariance under a larger set of transformations: conformal transformations. CFTs have an axiomatic mathematical definition [Sch08] and can describe natural phenomena such as second-order phase transitions, as well as complex theories such as string theory. The $2D$ Ising model, well known in statistical mechanics as well, is an example of CFT that researchers have completely solved [BPZ84].

The addition of conformal symmetry in CFTs is more than a simple restriction: it unlocks many tools and properties and puts additional constraints on the theory. The Operator Product Expansion (OPE) lets us write a product of operators as an infinite sum of individual operators and defines an associative algebra. In general QFTs, the OPE is an asymptotic expansion with a vanishing radius of convergence, while in CFTs this expansion is a convergent series with a strictly positive radius of convergence [Pol98], [PRER12].

Observables in QFTs are correlation functions between local operators inserted in points in space

$$\langle \phi(x_1) \cdots \phi(x_n) \rangle = \frac{1}{Z} \int [d\phi] \phi(x_1) \cdots \phi(x_n) e^{-S[\phi]} \tag{1.1}$$

where $Z = \langle 1 \rangle$ is the partition function while $[d\phi]$ and $S[\phi]$ are respectively a measure over fields and the *action* of the field, both supposed invariant under conformal transformations. Additionally, $e^{-S[\phi]}[d\phi]$ acts as a probability distribution.

Conformal symmetry fixes completely 2 and 3-point correlators. Higher-order correlators, on the other hand, are not fixed by conformal invariance and are dynamical quantities that vary from theory to theory. Thanks to the OPE, to calculate higher-order correlators we only need to look at 4-point functions and generalize the approach [FGG73]. If we expand the products $\phi(x_1)\phi(x_2)$ and $\phi(x_3)\phi(x_4)$, the correlator is written as a series of two-point functions of the form $\sum_i C_i^2 \langle \phi_{\Delta_i}(y) \phi_{\Delta_i}(z) \rangle$. Switching x_1 with x_3 leads to the same function but now with a different expansion. This equality between functions is an additional constraint that we have put on the CFT and is called the *Conformal bootstrap equation* [Pol74]. We can write the equation in the form

$$\sum_i C_i^2 F_{\Delta_i, s_i}(z, \bar{z}) = 0 \tag{1.2}$$

where the $F_{\Delta_i, s_i}(z, \bar{z})$ are defined in terms of known functions called *conformal blocks*. The equation has to be satisfied for all points in the complex plane except the half

lines $(-\infty, 0]$ and $[1, \infty)$, with unknowns being the squared OPE coefficients C_i^2 and the operators' scaling dimensions Δ_i and spins s_i.

Solving this equation is a hard task. Previous approaches consist of numerical studies on the conformal bootstrap equation as a way to exclude possible values for the scaling dimensions [RRTV08] using semidefinite programming [SD15]. Reviews of these methodologies can be found in [PRV19], [SD16]. This found application in various CFTs, most remarkably the $3D$ Ising model [KPSD14]. Most recent approaches include Monte Carlo [LVS22] and Reinforcement Learning (RL) techniques [KPN22a, KPN22b, KNPR23] with the latter being the basis of our work.

RL [SB18] is a technique in Machine Learning (ML) that involves the interaction between two components: the agent and the environment. At each step, the agent chooses an action and receives a reward and some kind of information on the environment. The objective is to find a strategy, or *policy*, that maximizes the cumulative reward in the long run. Recent algorithms can deal with complex problems and large environments using neural networks as policy and value-function estimators. Soft Actor-Critic (SAC) [HZAL18] is an example that uses 4, but has a particular addition to it. SAC includes in its objective function an entropy term on the stochastic policy to regulate how much it should explore the environment compared to the best policy it has found up to that point.

Attempts to solve the conformal bootstrap equation with RL use the Soft Actor-Critic (SAC) algorithm. In fact, with SAC it is possible to find an approximate solution to the conformal bootstrap equation for the $2D$ Ising model and other CFTs [KPN22a, KPN22b, KNPR23]. The critical part of this approach is translating the CFT setting and the conformal bootstrap equation as an agent-environment framework. This is done by truncating the equation up to a maximum number of operators or a maximum value for the operators' scaling dimensions and selecting a set of N_z points in the complex plane where the conformal bootstrap equation is evaluated. At each step, the agent performs an action by updating a couple of points (Δ_i, C_i^2), while the reward and the information received are based on the evaluation of the equation on the selected points and the current CFT data. This way, the agent learns how to move in the landscape of the conformal bootstrap equation, following the optimal route toward a solution.

We validate the technique on the $2D$ Ising model where an analytical solution is available. We show that if the search is kept completely free on all 22 unknowns the problem is very complex and the results are sometimes unexpected. By fixing an increasing number of scaling dimensions, the problem becomes easier and more approachable with the results being more coherent with the theoretical expectations. Additionally, by giving Δ values as input to the algorithm we put ourselves in a setting closer to the one of interest: the $1D$ defect CFT on the $\frac{1}{2}$-BPS Wilson line in a $4D$ theory known as $\mathcal{N} = 4$ super Yang-Mills.

This theory is related to string theory and studies particles and their fields, such as bosons and fermions, in a supersymmetric 4 dimensional setting with conformal with conformal symmetry. The $1D$ defect CFT has one important parameter describing interactions: the coupling constant g. In this theory, no analytical solution to the conformal bootstrap equation is available, although the scaling dimensions of the first 10 operators are known with good precision [CGJP22a] using integrability [GKP98, BAA+11, DKN+19, Lip94, FK95, MZ03]. Furthermore, the theory has 2 additional constraints we can impose on the integrals of functions of the conformal

blocks [CGJP22a, DK06a, CGJP22b], which can increase the precision of the results. These constraints were used to obtain bounds on the first three squared OPE coefficients [CGJP22a]. We remark that no previous application of RL on this theory is available in the literature and that is not straightforward to apply a working RL setting to a new problem in which the environment and the reward are different.

Once we determine the best way to include the 2 integral constraints into the reward, we focus on finding the OPE coefficients that solve the conformal bootstrap equation with the scaling dimensions fixed to the provided values. We can also fix the first OPE coefficient as the available bounds on it are very narrow while, for the second and the third coefficients, the situation is different and highly depends on the coupling constant g. We show that for small values of g the results obtained are very precise while being not very coherent with previous research, while larger values of g produce interesting results but lack precision whenever two operators have a very similar scaling dimension. As we will see, this is an expected consequence of the conformal blocks being analytical functions of Δ.

The rest of this thesis is organized as follows.

In Chapter 2, we define all the tools necessary to introduce CFTs in a mathematical and axiomatic way, studying conformal transformations and their action on the operators of the theory. We derive the form of the 2 and 3-point correlators using conformal symmetry, and we obtain the conformal bootstrap equation from the OPE.

Chapter 3 introduces ML and RL concepts, from optimization up to recent algorithms involving value function approximations and stochastic policies. We finally introduce Soft Actor-Critic (SAC), which is the main algorithm of the bootstrap stochastic optimization.

In Chapter 4 we define the environment and the RL setting to solve the conformal bootstrap equation, going through all the parameters and specifications. We also describe the two theories of our interest: the $2D$ Ising model, used as a benchmark of the approach, and the $1D$ defect CFT, the main objective of this work.

Finally, chapter 5 presents the results obtained by running the algorithm multiple times and selecting the best runs, differentiating small coupling $g \leq \frac{1}{2}$ from strong coupling $g \geq 1$. We further discuss the critical points of the method and possible causes of the loss in precision for some values of g.

Chapter 2

Conformal Field Theories

2.1 From Quantum Mechanics to Quantum Fields

In this first section, we introduce the formalisms of Quantum Mechanics (QM) and Quantum Field Theories (QFT), which will form the basis of the theories we will later analyze in depth: the Conformal Field Theories (CFT). In particular, we focus on:

- States, observables and operators in QM, the elements for a mathematical description of physical systems with a finite number of degrees of freedom and the measurements we can perform on them.
- An axiomatic definition of QFT and the generalizations of QM concepts to infinite degrees of freedom.
- The mathematical tools and objects to study QFTs such as the Operator Product Expansion (OPE), a very important tool that will turn out to be fundamental for further studies on CFTs.

QM is the backbone of modern physics which led to the explanation of many phenomena and aspects of nature in ways we could not comprehend with Classical Mechanics, such as atoms, the nature of light and the tunnel effect. QM requires us to abandon the deterministic point of view on the study of natural phenomena which had been dominant since the first historical approaches in ancient times.

In Classical Mechanics our measurements of physical systems can be arbitrarily good: any uncertainty is related to our instruments or human error but there is no theoretical limit to our precision. In QM, on the other hand, uncertainty and randomness are intrinsic to the theory and there is no possible way of increasing the precision of a measurement beyond a certain limit set by nature itself, other than shifting the error from one quantity to another. A famous example of this is Heisenberg's uncertainty principle: if σ_x and σ_p are the standard deviations of the variables representing respectively the position and momentum of a particle, we have that $\sigma_x \sigma_p \geq \frac{\hbar}{2}$, where $\hbar = \frac{h}{2\pi}$ and h is the Plank's constant. This inequality shows in a mathematical form that an increase in precision for a physical quantity often requires sacrificing precision in another.

Another fundamental difference that characterizes QM is quantization: in some physical systems, quantities like energy and angular momentum of particles cannot take any continuous as in Classical Mechanics. Instead, only a discrete set of values turns out to be physically realized. This is true not only for the state of a single particle

but also for interactions: exchanges of energy between systems often happen in packets of small but not continuous quantities, called quanta.

2.1.1 Formalism of Quantum Mechanics

We will now review the mathematical description of QM, starting from a fundamental physical object: the state of a particle, which is represented by an element of a Hilbert space called the **wavefunction**.

States

We will consider the tridimensional space $\mathbb{R}^3$ to be the space where the particles live. The reference Hilbert space to describe the physical states of the particle is $\mathcal{H} = L^2(\mathbb{R}^3; \mathbb{C})$ that is the space of square-integrable functions on $\mathbb{R}^3$ with values in the complex plane $\mathbb{C}$, that we will write simply as $\mathcal{H} = L^2(\mathbb{R}^3)$.

Definition 2.1. A **state** of a physical system is represented by a wavefunction $\psi(\boldsymbol{x}, t)$, that is a complex-valued function that at any time t is an element of the Hilbert space $\mathcal{H} = L^2(\mathbb{R}^3)$ and is normalized with norm 1.

The inner product that makes $\mathcal{H}$ a Hilbert space, with the usual QM notation of $|\psi\rangle = \psi(\boldsymbol{x}, t)$, is defined as

$$\langle\psi|\phi\rangle = \int_{\mathbb{R}^3} \psi(\boldsymbol{x})^* \phi(\boldsymbol{x}) \mathrm{d}x^3 \tag{2.1}$$

where $\psi(\boldsymbol{x}, t)^*$ denotes the complex conjugate. For simplicity, we will sometimes omit the time variable t and when two or more functions are used together we assume them to be evaluated at the same time.

In the above definition, we added a requirement on the norm of the wavefunction. As a consequence, the space of admissible states is the projective space $\mathbb{P}(\mathcal{H})$, with states being equivalent if they differ by a complex factor. When thinking of wavefunction as elements of $\mathcal{H}$, the states we are interested in are therefore those with $\|\psi\|^2 = \langle\psi|\psi\rangle < \infty$ since we can always normalize it to have norm 1.

Since the space of states is a Hilbert space we can also consider linear combinations of states: if we have $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$ and $|\psi\rangle, |\phi\rangle$ are admissible states, also $\alpha|\psi\rangle + \beta|\phi\rangle$ is an admissible state.

Physically the wavefunction contains all the information describing the physical state of a particle and, in particular, the squared norm of its value evaluated in position $\boldsymbol{x}$ at time t corresponds to a probability density $|\psi(\boldsymbol{x}, t)|^2 = p(\boldsymbol{x}, t)$ for which

$$\mathbb{P}[\text{particle in } M \subseteq \mathbb{R}^3 \text{ at time t}] = \int_M |\psi(\boldsymbol{x}, t)|^2 \mathrm{d}x \tag{2.2}$$

The normalization requirement makes more sense now since it is equivalent to the probability of the particle being anywhere in the whole $\mathbb{R}^3$.

The time evolution of the states is governed by the **Schrodinger's equation**:

$$i\hbar \frac{\partial \psi}{\partial t} = \hat{H}\psi \tag{2.3}$$

where $\hat{H}$ is a Hermitian operator on the Hilbert space $L^2(\mathbb{R}^3)$ and is called the **Hamiltonian**, which takes the same form of the classical Hamiltonian but with the scalar position and momentum replaced with the corresponding operators that we will shortly see. The Hamiltonian contains the laws of physics we are considering for the evolution of the particle, while Schrodinger's equation describes the evolution of physical systems. The formal solution can be written as

$$\psi(\boldsymbol{x}, t) = e^{-\frac{i}{\hbar}\int \hat{H}(t')dt'}\psi(\boldsymbol{x}, 0) \tag{2.4}$$

and, to be calculated, requires the diagonalization of the operator $\hat{H}$.

Notice how Schrodinger's equation is differential in the first order in contrast to the classical $\boldsymbol{F} = m\boldsymbol{a}$. This goes with the fact that a state in QM is completely described by just $\psi(\boldsymbol{x}, t)$ without the derivative $\dot{\psi}(\boldsymbol{x}, t)$.

Observables and Operators

In classical mechanics any function $f(\boldsymbol{x}, \boldsymbol{p})$ of the position $\boldsymbol{x}$ and momentum $\boldsymbol{p}$ can be considered as an observable. Example of observables are the energy $E(\boldsymbol{x}, \boldsymbol{p}) = \frac{\boldsymbol{p}^2}{2m} + V(\boldsymbol{x})$ or the angular momentum $\boldsymbol{L} = \boldsymbol{x} \times \boldsymbol{p}$. In QM, functions get replaced by operators:

Definition 2.2. An observable in Quantum Mechanics is a hermitian linear operator $\hat{O}$ acting on the Hilbert space $L^2(\mathbb{R}^3)$

where an hermitian operator satisfies the following relation: $\langle\phi|\hat{O}\psi\rangle = \langle\hat{O}^\dagger\phi|\psi\rangle$.

In QM, observables are operators acting on the underlying space of states. Example of basic operators are the position operator $\hat{\boldsymbol{x}}\psi(\boldsymbol{x}, t) = \boldsymbol{x}\psi(\boldsymbol{x}, t)$, the momentum operator $\hat{\boldsymbol{p}} = -i\hbar\nabla$, the angular momentum operator $\hat{\boldsymbol{L}} = -i\hbar\boldsymbol{x} \times \nabla$ and the energy operator $\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\boldsymbol{x})$. The energy operator is exactly the Hamiltonian for particles moving in a potential $V(\boldsymbol{x})$.

Definition 2.3. Given an operator $\hat{O}$, if there exist $\lambda \in \mathbb{C}, \psi(\boldsymbol{x}) \in L^2(\mathbb{R}^3)$ such that $\hat{O}\psi(\boldsymbol{x}) = \lambda\psi(\boldsymbol{x})$, then λ is an **eigenvalue**, $\psi(\boldsymbol{x})$ is a **eigenstate** or **eigenfunction** and the collection of the eigenvalues is called the **spectrum**.

Hermitian operators have very important properties for our purposes:

- We can construct a complete basis of an orthonormal eigenfunction of any hermitian operator $\hat{O}$: this means that any $\psi \in L^2(\mathbb{R}^3)$ can be written as an (infinite) linear combination of the eigenstates $\psi(\boldsymbol{x}) = \sum_{i=0}^{\infty} a_n\phi_n(\boldsymbol{x})$.
- All their eigenvalues λ_n are real, which makes them very good candidates to describe physical observables.
- If two eigenvalues are different $\lambda_n \neq \lambda_m$ then the corresponding eigenfunctions are orthogonal $\langle\phi_n|\phi_m\rangle = 0$.

One fundamental fact in QM is that the outcome of the measurement always lies in the spectrum of the corresponding operator. The second property of hermitian operators is therefore very important: all measurements give real outcomes. If we decompose a state into the orthonormal basis $\psi(\boldsymbol{x}) = \sum_{i=0}^{\infty} a_n\phi_n(\boldsymbol{x})$, the probability of the measurement being λ_n is given by the **Born rule**:

Definition 2.4. Let a measurable be described by a hermitian operator $\hat{O}$, with eigenvalues λ_n, eigenvectors $\phi_n(\boldsymbol{x})$ and each eigenspace being related to a projection operator $\hat{P}_n$. From the measurement of $\hat{O}$, the probability of obtaining the outcome λ_n is

$$\mathbb{P}[\lambda_n] = \langle\psi|\hat{P}_n|\psi\rangle \tag{2.5}$$

If all eigenspaces have dimension 1, the probability takes the form

$$\mathbb{P}[\lambda_n] = \sum_{k,l} a_k a_l \langle\phi_k|\hat{P}_n|\phi_l\rangle = |a_n|^2 \tag{2.6}$$

Since a physical state is normalized and the norm of $\psi(\boldsymbol{x})$ given its decomposition is $\sum_{n=0}^{\infty}|a_n|^2$ we are sure that $\sum_{n=0}^{\infty}\mathbb{P}[\lambda_n] = 1$, which means that the probabilities for every outcome sum up to 1. When a measurement is performed, the initial state $\psi(\boldsymbol{x})$ collapses into the eigenstate corresponding to the eigenvalue obtained $\psi(\boldsymbol{x}) \to \phi_n(\boldsymbol{x})$.

A useful mathematical relation is that the expected value of the outcome of a measurement can be written as $\langle\hat{O}\rangle_\psi = \langle\psi|\hat{O}\psi\rangle = \sum_{n=0}^{\infty}|a_n|^2\lambda_n$. For example, the average position of a particle in state $\psi(\boldsymbol{x})$ is given by $\langle\psi|\hat{\boldsymbol{x}}\psi\rangle = \int_{\mathbb{R}^3}\mathrm{d}x^3|\psi(\boldsymbol{x})|^2\boldsymbol{x}$ which is appropriate, recalling that $|\psi(\boldsymbol{x})|^2$ is the density probability of the particle being at $\boldsymbol{x}$.

An important concept related to operators and measurement is **commutation**.

Definition 2.5. The **commutator** of two operators $\hat{O}, \hat{M}$ is the operator defined as

$$[\hat{O}, \hat{M}] = \hat{O}\hat{M} - \hat{M}\hat{O} \tag{2.7}$$

acting as

$$[\hat{O}, \hat{M}]\psi(\boldsymbol{x}) = \hat{O}\hat{M}\psi(\boldsymbol{x}) - \hat{M}\hat{O}\psi(\boldsymbol{x}) \tag{2.8}$$

Assume we are measuring an observable associated with $\hat{O}$. After the measurement the initial state $\psi(\boldsymbol{x})$ collapses in a wavefunction $\phi_1(\boldsymbol{x})$ that is an eigenfunction of $\hat{O}$. If we then measure $\hat{M}$, the new state obtained afterwards $\phi_2(\boldsymbol{x})$ is often not an eigenfunction of $\hat{O}$ and the result of a subsequent measurement of $\hat{O}$ may produce a different outcome from the previous.

Being able to measure two observables simultaneously or subsequently without losing information is strictly related to the commutation relations of the operators. In particular, two observables $\hat{O}, \hat{M}$ can be measured simultaneously if and only if $[\hat{O}, \hat{M}] = 0$ and this happens if and only if $\hat{O}$ and $\hat{M}$ are simultaneously diagonalizable.

As an example, the commutation relations of the position $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \hat{x}_3)$ and the momentum operator $\hat{\boldsymbol{p}} = (\hat{p}_1, \hat{p}_2, \hat{p}_3)$ are given by

$$[\hat{x}_i, \hat{x}_j] = [\hat{p}_i, \hat{p}_j] = 0 \;\text{ and }\; [\hat{x}_i, \hat{p}_j] = i\hbar\delta_{ij} \tag{2.9}$$

The position and momentum operators do not commute and the Heisenberg's uncertainty principle can be proven to be a consequence of these relations.

2.1.2 Quantum Field Theory

Quantum Mechanics deals with a finite number of degrees of freedom while, in general, we may want to study quantized systems where the degrees of freedom are infinite.

Classical fields

In classical theories, a distribution of degrees of freedom for each point in space can be described as a field:

Definition 2.6. A **field** is a quantity defined at every point of time and space $(\boldsymbol{x}, t)$, written as $\phi_a(\boldsymbol{x}, t)$ where a is a label.

The dynamics of a single particle are described by a finite number of coordinates $q_a(t)$, indexed by a, while in field theory we consider both a and $\boldsymbol{x}$ as labels or indexes. Classical examples of fields are the electric and magnetic fields $\boldsymbol{E}(\boldsymbol{x}, t), \boldsymbol{B}(\boldsymbol{x}, t)$, both of which are 3-dimensional and governed by Maxwell's equations. Using the Einstein notation and the 4-component version of the field in spacetime we can define the electromagnetic potential as $A^\mu(\boldsymbol{x}, t) = (\phi, \boldsymbol{A})$ with $\mu = 0, 1, 2, 3$ and the 0 component being the time component. Electric and magnetic fields are then obtained by

$$\boldsymbol{E} = -\nabla\phi - \frac{\partial \boldsymbol{A}}{\partial t} \text{ and } \boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{2.10}$$

from which the first two Maxwell's equations $\nabla \cdot \boldsymbol{B} = 0$ and $\frac{d\boldsymbol{B}}{dt} = -\nabla \times \boldsymbol{E}$ follow immediately.

The dynamics of fields are governed by the Lagrangian:

Definition 2.7. A **Lagrangian** is a function of $\phi_a(\boldsymbol{x}, t), \partial_\mu\phi_a(\boldsymbol{x}, t)$ and $\nabla\phi_a(\boldsymbol{x}, t)$ in the form

$$L(t) = \int_{\mathbb{R}^3} \mathrm{d}x^3 \mathcal{L}(\phi_a, \partial_\mu\phi_a) \tag{2.11}$$

where $\mathcal{L}$ is the **Lagrangian density**, often called Lagrangian as well. The **action** is defined as

$$S = \int_{t_1}^{t_2} \mathrm{d}t \int_{\mathbb{R}^3} \mathrm{d}x^3 \mathcal{L} = \int \mathrm{d}x^4 \mathcal{L} \tag{2.12}$$

From the principle of least action, we can obtain the usual Euler-Lagrangian equations for the motion of fields, well known in the theory of Calculus of Variations.

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\mu\phi_a)} \right) - \frac{\partial \mathcal{L}}{\partial \phi_a} = 0 \tag{2.13}$$

Going back to the example of the electromagnetic field we consider the Lagrangian

$$\mathcal{L} = -\frac{1}{2}(\partial_\mu A_\nu)(\partial^\mu A^\nu) + \frac{1}{2}(\partial_\mu A^\mu)^2 \tag{2.14}$$

from which we obtain

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\mu\phi_a)} \right) = -\partial_\mu(\partial^\mu A^\nu - \partial^\nu A^\mu) \equiv -\partial_\mu F^{\mu\nu} \tag{2.15}$$

and, using (2.13), we obtain the remaining two Maxwell's equations $\nabla \cdot \boldsymbol{E} = 0, \frac{\partial \boldsymbol{E}}{\partial t} = \nabla \times \boldsymbol{B}$.

Our next step is to describe the formalism behind an axiomatic definition of Quantum Fields, starting from the field operators or Quantum Fields.

Field Operators

To build a mathematical framework for Quantum Field Theories we need to first define field operators and recall some common definitions. Let $\mathcal{H}$ be a Hilbert space.

Definition 2.8. An **operator** is pair (A, D) consisting of a subspace $D = D_A \subset \mathcal{H}$ and a $\mathbb{C}$-linear mapping $A : D \to \mathcal{H}$. A is densely defined if D_A is dense in $\mathcal{H}$. $\mathcal{O}(\mathcal{H})$ is the set of all densely defined operators on the Hilbert space.

We call $\mathcal{SO}(\mathcal{H})$ the set of all self-adjoint operators on $\mathcal{H}$.

Definition 2.9. The **Schwartz space** is the space of rapidly decreasing smooth functions, that is the complex vector space of all functions $f : \mathbb{R}^d \to \mathbb{C}$ with continuous partial derivatives of any order satisfying

$$|f|_{p,k} = \sup_{|\alpha|\leq p} \sup_{x\in\mathbb{R}^n} |\partial^\alpha f(x)|(1+|x|^2)^k < \infty \tag{2.16}$$

for all $p, k \in \mathbb{N}$ where α is a multi-index.

Definition 2.10. A **tempered distribution** T is a linear functional $T : \mathcal{S} \to \mathbb{C}$ continuous with respect to all the seminorms $|f|_{p,k}$.

We are now ready to give a formal definition of a quantum field

Definition 2.11. A **field operator** or **quantum field** is an operator-valued distribution

$$\phi : \mathcal{S}(\mathbb{R}^d) \to \mathcal{O}(\mathcal{H}) \tag{2.17}$$

such that there exist a dense subspace $D \subset \mathcal{H}$ satisfying:

- for each $f \in \mathcal{S}(\mathbb{R}^d)$ the domain of definition of $D_{\phi(f)}$ contains D.
- The induced map $\mathcal{S}(\mathbb{R}^d) \to \mathrm{End}(D), f \mapsto \phi(f)|_D$ is linear.
- For each $v \in D$ and $w \in \mathcal{H}$ the assignment $f \mapsto \langle w, \phi(f)(v)\rangle$ is a tempered distribution.

Field operators are the analogs of a classical field in the quantum framework. There is still one very important concept in physics we have not presented yet that is fundamental for QFTs: **relativistic invariance**. We will not go into the detail of the theory of relativity, nor how it is applied to QM and QFT, but we will briefly describe the group of transformations and the invariance relations needed to define the axiomatic QFT.

We will now work within the framework of the Minkowski space $M = \mathbb{R}^{1,d-1}$, that is the space $\mathbb{R}^d$ equipped with the Lorentz metric

$$x^2 = \langle x, x\rangle = x^0x^0 - \sum_{j=1}^{d-1} x^j x^j = g^{1,d-1}_{\mu\nu} x^\mu x^\nu \tag{2.18}$$

Definition 2.12. Two subsets U, V of M are **space-like separated** if for any $x \in U, y \in V$ we have $(x-y)^2 < 0$ that is if

$$(x^0 - x^0) < \sum_{j=1}^{d-1} (x^j - y^j)^2 \tag{2.19}$$

The **forward cone** is $C_+ := \{x \in M : \langle x, x\rangle \geq 0, x^1 \geq 0\}$ with the **causal order** given by $x \geq y \iff x - y \in C_+$.

These concepts are very important for classical particles and fields in the relativistic setting since they are related to the causal ordering of events. In particular, the law of physics and the action of fields must be invariant under relativistic transformations, which are described by the Poincaré group $P(1, d-1)$ of transformations. The Poincaré group is the group of isometries of the Minkowski space with the Lorentz metric above and includes:

- Translations in time and space, identified by the commutative group $\mathbb{R}^d$.
- The Lorentz group L of rigid rotations in space and relativistic boosts, which is isomorphic to the component containing the identity of $SO(1, d-1)$ and corresponds to the group of linear transformations that preserve the space-time causal structure and separateness.

$P(1, d-1)$ is actually the semi-direct product of the two components $P \simeq L \ltimes \mathbb{R}^d$. An important remark to make is that the Lorentz group is related to the concept of **spin**, which is a characteristic of particles and, in general, of operators. Spin is related to the angular momentum and magnetic properties of particles but, mathematically, we are interested in the fact operators can have a spin number l and that there exist a set of transformations represented by a group called $\mathrm{Spin}(1, d-1)$. This group describes how operators with spin transform under some transformations such as rotations and is actually the universal cover of the group $SO(1, d-1)$ we will later see.

The Poincaré group acts on $\mathcal{S}(\mathbb{R}^n)$ from the left as

$$(q, \Lambda) f(x) = f(\Lambda^{-1}(x - q)) \tag{2.20}$$

where $(q, \Lambda) \in L \ltimes \mathbb{R}^d$.

Without going in-depth with representation theory we assume to have a mapping

$$U : P(1, d-1) \to U(\mathcal{H}), (q, \Lambda) \mapsto U(q, \Lambda) \tag{2.21}$$

where $U(\mathcal{H})$ is the set of unitary transformations on the space of states. This mapping associates each transformation in the group to a unitary operator acting on states. An important fact is that we can write

$$U(q, 1) = \exp(i(q^0 P_0 - q^1 P_1 - \ldots - q^{d-1} P_{d-1})) \tag{2.22}$$

for $q \in \mathbb{R}^{1,d-1}$ and the P^μ are self-adjoint operators on $\mathcal{H}$. P_0 is considered to be the energy operator while the P_j operators represent the momentum operators as in QM.

Wightman axioms

Definition 2.13. A **Wightman quantum field theory** (Wightman QFT) in dimension d consists of the following elements:

- The space of states, that is the projective space $\mathbb{P}(\mathcal{H})$ of a separable Hilbert space $\mathcal{H}$.
- The vacuum vector $\Omega \in \mathcal{H}$ of norm 1.
- A unitary representation $U : P \to U(\mathcal{H})$ as above.

- A collection of field operators $\phi_a, a \in I$

$$\phi_a : \mathcal{S}(\mathbb{R}^d) \to \mathcal{O}(\mathcal{H}) \tag{2.23}$$

with a dense subspace $D \subset \mathcal{H}$ as common domain, which means that the domain $D_a(f)$ of ϕ_a contains D for all $a \in I, f \in \mathcal{S}(\mathbb{R}^d)$. Moreover, D must contain Ω.

These elements satisfy the following axioms:

Axiom W1 (Covariance). A Wightman QFT is assumed to have the following properties:

- Ω is P-invariant, that is for every (q, Λ) we have

$$U(q, \Lambda)\Omega = \Omega \tag{2.24}$$

and D is invariant under P, that is $U(q, \Lambda)D \subset D$.

- The common domain is invariant in the sense that $\phi_a(f)D \subset D$ for all $f \in \mathcal{S}(\mathbb{R}^n)$ and $a \in I$.

- The actions on $\mathcal{H}$ and $\mathcal{S}(\mathbb{R}^n)$ are *equivariant* where P acts on $\mathrm{End}(D)$ by conjugation. That is, on D we have

$$U(q, \Lambda)\phi_a(f)U(q, \Lambda)^\dagger = \phi_a((q, \Lambda)f) \tag{2.25}$$

for all f and all conformal transformations.

Axiom W2 (Locality). $\phi_a(f)$ and $\phi_b(g)$ commute on D if the supports of $f, g \in \mathcal{S}(\mathbb{R}^n)$ are space-like separated

$$\phi_a(f)\phi_b(g) - \phi_b(g)\phi_a(f) = [\phi_a(f), \phi_b(g)] = 0 \tag{2.26}$$

Axiom W3 (Spectrum condition). The joint spectrum of the operators P_j defined above is contained in the forward cone C_+.

Axiom W4 (Uniqueness of the Vacuum). The only vectors in $\mathcal{H}$ left invariant by the translations $U(q, 1)$ are the scalar multiples of the vacuum Ω.

As we can see from the axioms, the Poincaré group and the covariance under relativistic transformations are key parts of the definitions of QFTs.

From now on, we will write $\phi_a(f)$ as an operator-valued function on the support of f writing simply $\phi_a(x) = \phi_a(f(x))$. This is purely a formal way of writing field operators since in reality, as shown in [Sch08], quantum fields cannot be represented as functions. With this new notation, we can write more clearly the third point of Axiom W1 (Covariance) as

$$U(q, \Lambda)\phi_a(x)U(q, \Lambda)^\dagger = \phi_a(\Lambda x + q) \tag{2.27}$$

Before we introduce conformal transformations and Conformal Field Theories, let us review some objects we introduced for QM and classical fields in the framework of QFTs, starting from states.

In order to construct the states, we need to foliate the space-time. For QFTs in the Minkowski space $\mathbb{R}^{1,3}$ we can foliate space-time by considering surfaces of equal time, also called leaves.

$$L(t_0) = \{x \in \mathbb{R}^{1,3} : x^0 = t_0\} \simeq \mathbb{R}^3 \tag{2.28}$$

Each time slice has a corresponding Hilbert space of states defined, in our case $L^2(\mathbb{R}^3)$. Particles in QFT are local excitations of elementary fields (such as the vacuum state $|0\rangle$), written as $|\psi\rangle = \phi(x)\,|0\rangle$. This operation of adding a field operator to a state is often called **insertion**. For example, photons are excitations given by the insertion of a quantized electromagnetic field.

The correlation between states in the same leaf is written simply as $\langle\phi|\psi\rangle$. To correlate operators living in different time slices we need to evolve one of them with a translation in time, which is represented by the unitary operator $U = e^{-iP_0\Delta t}$, where Δ_t is the time difference between the two slices. Then, the correlation becomes

$$\langle\psi|U|\phi\rangle \tag{2.29}$$

One of the main objects of QFT is the n-point correlator function:

Definition 2.14. Given a measure over fields $[d\phi]$ the **n-point correlator function** is defined as

$$\langle\phi_1(x_1)\cdots\phi_n(x_n)\rangle = \frac{1}{Z}\int[d\phi]\phi_1(x_1)\cdots\phi_n(x_n)e^{-S[\phi]} \tag{2.30}$$

where Z is a normalization constant $Z = \langle 1\rangle$ and is called the **partition function**.

Correlators can be thought of as the observables of QFT and from them one can obtain the probabilities of particles interacting.

A very important tool to calculate correlators is the Operator Product Expansion (OPE). Consider a state obtained from two insertions

$$|\psi\rangle = \phi(x)\phi(0)\,|0\rangle \tag{2.31}$$

From translational invariance, we can assume one of them to be inserted at the origin. In any Lorentz-invariant QFT, the operator product $\phi(x)\phi(0)$ in the short distance limit $x \to 0$ can be approximated as a sum of local operators of the form

$$\phi(x)\phi(0) \simeq \sum_n C_n(x^2)x^{\mu_1}\cdots x^{\mu_l}\phi^{(n)}_{\mu_1\ldots\mu_l}(0) \tag{2.32}$$

where C_n depends only on x^2 and l is the spin number of operator $\phi^{(n)}$. The problem with this tool is that, in general, it is just an asymptotic expansion, meaning that it can have a vanishing radius of convergence. This is much weaker than having a convergent sum which, as we will see, is the case for CFTs.

Lastly, QFTs have a Lagrangian formalism similar to classical fields. For example, Lagrangians typically are polynomials of fields and their derivatives and they can be thought of as composed of two terms

$$\mathcal{L} = \mathcal{L}_{\text{free}} + g\mathcal{L}_{\text{int}} \tag{2.33}$$

where $\mathcal{L}_{\text{free}}$ is the free term which is quadratic in the fields, making the underlying theory exactly solvable, and $g\mathcal{L}_{\text{int}}$ is the interaction term. The constant g is called **coupling constant** and determines the relation between the two terms. Usually, these are very complicated objects and we can derive expansions of the Lagrangian for small values of g, the weak coupling limit, while it is harder to access the dynamics of systems with strong coupling. As a final example, the Lagrangian density for quantum electrodynamics can be written as

$$\mathcal{L} = \bar{\psi}(i\hbar c\gamma^\mu - mc^2)\psi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} - ec\bar{\psi}\gamma^\mu A_\mu\psi \tag{2.34}$$

Without explaining the quantities in the Lagrangian, we can see the two terms above being

$$\begin{aligned} \mathcal{L}_{\text{free}} &= \bar{\psi}(i\hbar c\gamma^\mu - mc^2)\psi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} \\ \mathcal{L}_{\text{int}} &= c\bar{\psi}\gamma^\mu A_\mu \psi \end{aligned} \tag{2.35}$$

where the coupling constant is $g = e$, the elemental charge of the electron.

2.2 Conformal Transformations

Conformal Field Theories are a special kind of QFTs that can be used to describe phase transitions and critical phenomena, such as ferromagnetic transition or complex theories such as string theories. In the modern understanding of QFTs, CFTs play a fundamental role: every QFT can be thought of as the result of a flow between an ultraviolet CFT, which describes the theory at very high energies, and an infrared CFT describing the theory at low energies. The key property of CFTs is that, in addition to the usual Poincaré invariance, they also enjoy scale invariance. In most theories, the latter is further enhanced to invariance under conformal transformations, that is the group of transformations that leaves the metric unchanged up to a local multiplicative factor.

In this section we will study conformal transformations and their group, starting with their definition and derivation, followed by their characterization in general dimension and for $d = 2$, which has particular features.

Definition 2.15. A **semi-Riemannian manifold** is a pair $(M,\ g)$ consisting of a smooth manifold M of dimension d and a smooth tensor field g which assigns each point $a \in M$ to a non-degenerate bilinear form on the tangent space T_aM

$$g_a : T_aM \times T_aM \to \mathbb{R} \tag{2.36}$$

Consider a chart $\phi : U \to V$ with $U \subset M$ open and $V \subset \mathbb{R}^D$ and the local coordinates of the chart $x^1, x^2, \ldots, x^d$. Given two tangent vectors $X = X^\mu\partial_\mu, Y = Y^\mu\partial_\mu \in T_aM$ we can write, using the Einstein's notation:

$$g_a(X, Y) = g_{\mu\nu}(a)X^\mu Y^\nu \tag{2.37}$$

where $\partial_\mu = \frac{\partial}{\partial x^\mu}$ is the basis on the tangent space induced by chart ϕ.

The matrix $g_{\mu\nu}(a)$ is non-degenerate and symmetric for all $a \in U$ and, since g_a is smooth, it is differentiable with respect to a and smooth in the local coordinates. The only difference from Riemannian manifolds is that g_a is not strictly positive definite.

For us, the most important example of a semi-Riemannian manifold is the space $\mathbb{R}^{p,q} = (\mathbb{R}^{p+q}, g^{p,q})$ where

$$g^{p,q}(X, Y) = \sum_{i=1}^{p} X^iY^i - \sum_{i=p+1}^{p+q} X^iY^i \tag{2.38}$$

For example, $\mathbb{R}^{1,3}$ (or $\mathbb{R}^{3,1}$) is the usual Minkowski space for space-time we have already seen, used to describe relativity and quantum fields. $\mathbb{R}^{1,1}$ is the Minkowski plane and $\mathbb{R}^{2,0}$ is the usual Euclidean plane.

We are ready to define conformal transformations:

Definition 2.16. Let (M, g) and (M', g') two semi-Riemannian manifolds of same dimension n and let $U \subset M, V \subset M'$ be open subsets. A **conformal transformation** is a smooth mapping $\phi : U \to V$ of maximum rank such that there exist a smooth function $\Omega : U \to \mathbb{R}_+$ such that

$$\phi^* g' = \Omega^2 g \tag{2.39}$$

where $\phi^* g'(X, Y) = g'(T\phi(X), T\phi(Y))$ is the pull-back of the bilinear form and $T\phi : TU \to TV$ is the tangent map of ϕ. Ω is called the **conformal factor**.

This definition means that conformal transformations are the transformations that preserve the metric up to a local scale factor, which depends only on the point. In local coordinates of M, M' the above definition becomes

$$(\phi^* g')_{\mu\nu}(a) = {g'}_{\alpha\beta}(\phi(a))\partial_\mu \phi^\alpha \partial_\nu \phi^\beta \tag{2.40}$$

which leads to the fact that a transformation is conformal if and only if

$$({g'}_{\alpha\beta} \circ \phi)\partial_\mu \phi^\alpha \partial_\nu \phi^\beta = \Omega^2 g_{\mu\nu} \tag{2.41}$$

Our next objective is to identify and classify conformal transformations between two open subsets U, U' of the space $\mathbb{R}^{p,q}, p + q = d > 1$. Recall that if X is a smooth vector field we can define a differential equation for smooth curves γ given by $\dot{\gamma} = X(\gamma)$.

Definition 2.17. The **local one-parameter group** $(\phi_t^X)_{t\in\mathbb{R}}$ corresponding to X is the solution to

$$\frac{d}{dt}(\phi^X(t, a)) = X(\phi^X(t, a)) \tag{2.42}$$

with initial condition $\phi^X(0, a) = a$. For each point $a \in U$ we will consider $\phi^X(t, a)$ as the maximal solution to the above differential equation. For a fixed t, $\phi_t^X(a)$ is a local diffeomorphism.

A conformal transformation has max rank in the open set U and is therefore invertible for each $a \in U$. By the inverse mapping theorem, conformal transformations are always locally invertible but, in general, not globally!

From now, we will assume that the metric is $g_a = g_a^{p,q}$ on $\mathbb{R}^{p,q}$, which does not depend on the point a and is therefore constant.

Definition 2.18. A vector field X on $U \subset \mathbb{R}^{p,q}$ is called a **conformal Killing vector field** if ϕ_t^X is conformal for all t in a neighborhood of 0.

Theorem 2.1. *Let $U \subset \mathbb{R}^{p,q}$ open, $g = g^{p,q}$ and X a conformal Killing field with coordinates*

$$X = (X^1, \ldots, X^n) = X^\mu \partial_\mu \tag{2.43}$$

with respect to the cartesian coordinates of $\mathbb{R}^n$. Then there exist a smooth function $\kappa : U \to \mathbb{R}$ such that

$$X_{\mu,\nu} + X_{\nu,\mu} = \kappa g_{\mu\nu} \tag{2.44}$$

with the notations $f_{,\nu} = \partial_\nu f$ and $X_\mu = g_{\mu\nu} X^\nu$.

As for conformal transformation we also have a conformal Killing factor:

Definition 2.19. A smooth function $\kappa : U \subset \mathbb{R}^{p,q} \to \mathbb{R}$ is called a **conformal Killing factor** if there is a conformal Killing field such that

$$X_{\mu,\nu} + X_{\nu,\mu} = \kappa g_{\mu\nu} \tag{2.45}$$

Theorem 2.2. *$\kappa : U \to \mathbb{R}$ is a conformal Killing factor if and only if*

$$(n-2)\kappa_{,\mu\nu} + g_{\mu\nu}\nabla_g \kappa = 0 \tag{2.46}$$

*where $\nabla_g = g^{\mu\nu}\partial_\mu\partial_\nu$ is the **Laplace-Beltrami** operator for $g = g^{p,q}$.*

2.2.1 Classification of conformal transformations

Starting from equation (2.46) one can derive all the possible conformal transformations and classify the most simple ones. We will consider three cases in the discussion, depending on the considered space.

$\mathbb{R}^{p,q}$ with $p + q = n > 2$

From equation (2.46) one can derive that, in local coordinates x, there exist constants α_μ such that $\kappa(x) = \lambda + \alpha_\mu x^\mu$.

With some more calculations depending on the values of λ and the constants α_μ one can obtain the following

Theorem 2.3. *Every conformal Killing vector field X on a connected open subset U of $\mathbb{R}^{p,q}$ with $p + q > 2$ is of the form*

$$X(x) = 2\langle x, b\rangle x^\mu - \langle x, x\rangle + \lambda x + c + \omega x \tag{2.47}$$

with $b, c \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$ and ω is a matrix such that $\omega^T g^{p,q} + g^{p,q}\omega = 0$. $\langle\cdot,\cdot\rangle$ is the bilinear form given by g, that is

$$\langle x, y\rangle = g_{\mu\nu}x^\mu y^\nu \tag{2.48}$$

The conformal transformations in $\mathbb{R}^{p,q}$ with $p + q > 2$ can then be described as follows

Theorem 2.4. *Every conformal transformation $\phi : U \to \mathbb{R}^{p,q}$ with $p + q = n > 2$, on a connected open subset $U \subset \mathbb{R}^{p,q}$ is a composition of*

- *a translation $x \mapsto x + c$ with $c \in \mathbb{R}^n$.*
- *an orthogonal transformation $x \mapsto \Lambda x$ with $\Lambda \in O(p, q)$ where*

$$O(p, q) : \Lambda^T g^{p,q}\Lambda = g^{p,q}\} \tag{2.49}$$

- *a dilatation $x \mapsto e^\lambda x$ with $\lambda \in \mathbb{R}$*
- *a special conformal transformation of the form*

$$x \mapsto \frac{x - \langle x, x\rangle b}{1 - 2\langle x, b\rangle + \langle x, x\rangle\langle b, b\rangle} \tag{2.50}$$

with $b \in \mathbb{R}^n$.

These transformations are clearly not defined everywhere but, taking the compactification of $\mathbb{R}^{p,q}$, we can extend them everywhere and obtain the group of conformal transformations. A complete reference can be found in [Sch08] with the main result being

Theorem 2.5. *Let again our space be $\mathbb{R}^{p,q}$. The group of conformal transformations (on the conformal compactification of $\mathbb{R}^{p,q}$) is isomorphic to $O(p+1,q+1)/\{\pm 1\}$. The connected component containing the identity in this group is called the Conformal group Conf($\mathbb{R}^{p,q}$) and is isomorphic to $SO(p+1,q+1)$ or to $SO(p+1,q+1)/\{\pm\}$ if -1 is in the connected component of the identity in $O(p+1,q+1)$, for example when p and q are odd.*

Euclidean plane $\mathbb{R}^{2,0}$

The Euclidean plane can be identified as the complex plane $\mathbb{R}^{2,0} \simeq \mathbb{C}$ via the mapping $(x,y) \in \mathbb{R}^{2,0} \mapsto z = x + iy$. Using equation (2.41), a smooth map $\phi : U \to \mathbb{C}$ with $U \subset \mathbb{C}$ open if conformal with conformal factor Ω if and only if

$$u_x^2 + v_x^2 = \Omega^2 = u_y^2 + v_y^2 \neq 0, \quad u_x u_y + v_x v_y = 0 \tag{2.51}$$

where $u = \Re(\phi), v = \Im(\phi)$, $u_x = \partial_x u$ and the other definitions follow in a similar way. It is immediate to see that holomorphic and anti-holomorphic functions satisfy those relations because of the Cauchy-Riemann equations $u_x = v_y, u_y = -v_x$ (or $u_x = -v_y, u_y = v_x$ for anti-holomorphic). We also have that $\det D\phi \neq 0$.

On the other hand, using (2.51), one can also show the other implication. This means that

Theorem 2.6. *Every holomorphic function*

$$\phi = u + iv : U \to \mathbb{R}^{2,0} \simeq \mathbb{C} \tag{2.52}$$

on an open subset $U \subset \mathbb{R}^{2,0}$ with nowhere vanishing derivative is an orientation preserving conformal mapping with conformal Killing factor $\Omega^2 = u_x^2 + u_y^2 = det D\phi = |\phi'|^2$. Conversely, every conformal and orientation-preserving transformation $\phi : U \to \mathbb{R}^{2,0} \simeq \mathbb{C}$ is a holomorphic function.

A symmetric result can be obtained for anti-holomorphic functions. Finally, the group of conformal transformation on the compactification of $\mathbb{R}^{2,0}$ is isomorphic to $SO(3,1)$. Additionally, one can also show that the conformal group is also isomorphic to the group Möbius transformations

$$z \mapsto \frac{az+b}{cz+d} \tag{2.53}$$

One very important remark to be made is that, since every holomorphic and anti-holomorphic function is a local conformal transformation, there exists a much larger algebra of transformations in the 2 dimensional case, which is actually infinite-dimensional and is called **Virasoro algebra**. We will later see how this leads to the fact that conformal symmetry in $d = 2$ is much more powerful and should be studied separately. Notice that there is a big difference between the group of *global* conformal transformations, which is isomorphic to $SO(3,1)$ and is therefore finitely generated, and the algebra of *local* or *infinitesimal* conformal transformations, which is infinite-dimensional.

Minkowski plane $\mathbb{R}^{1,1}$

We can show the following

Theorem 2.7. *A smooth map $\phi = (u, v) : U \to \mathbb{R}^{1,1}$ on a connected open subset $U \subset \mathbb{R}^{1,1}$ is conformal if and only if*

$$u_x^2 > v_x^2, \text{ and } u_x = v_y, u_y = v_x \text{ or } u_x = -v_y, u_y = -v_x \tag{2.54}$$

In this case, both the algebra of local conformal transformations and the group of global conformal transformations are infinite dimensional, with the latter being

Theorem 2.8. *The conformal group for the Minkowski plane $\mathbb{R}^{1,1}$ is isomorphic to*

$$Conf(\mathbb{R}^{1,1}) \equiv Diff_+(\mathbb{S}) \times Diff_+(\mathbb{S}) \tag{2.55}$$

where $Diff_+(\mathbb{S})$ is the group of orientation preserving diffeomorphisms $f : \mathbb{R} \to \mathbb{R}$ with the topology of uniform convergence of f and its derivatives on compact subsets $K \subset \mathbb{R}$.

To make a connection between this case and theorem 2.5, one can show that the group $SO(2,2)/\{\pm 1\}$ is actually a subgroup of $\mathrm{Conf}(\mathbb{R}^{1,1})$.

2.3 Conformal Field Theories

In this section we will introduce the theories of our interest, following [Sch08], that is the Conformal Field Theories (CFT). We will start by giving the axiomatic definition, which is an extension of definition 2.13, then we will see how the conformal transformations act on states and operators. Finally, we will see the particularities of CFTs concerning correlators and the operator product expansion, which is more powerful than in general QFTs.

Conformal Field Theory is a special kind of Quantum Field Theory which, apart from the usual invariance under the Poincaré group of transformations, is also invariant under dilatations and special conformal transformations, making it invariant under the action of the whole conformal group. To apply the axiomatic definition to it, we assume to have a unitary representation of $SO(2,d)$ extending (2.21)

$$U : SO(2,d) \to U(\mathcal{H}), (q, \Lambda, b, \lambda) \mapsto U(q, \Lambda, b, \lambda) \tag{2.56}$$

where the generators (q, Λ, b, λ) refer to theorem 2.4. The Wightman axioms, especially the first, are now extended to include covariance under such transformations, so that

$$U(q, \Lambda, b, \lambda)\phi_a(f)U(q, \Lambda, b, \lambda)^\dagger = \phi_a((q, \Lambda, b, \lambda)f) \tag{2.57}$$

Using the same abuse of notation as before, writing $\phi_a(f)$ as an operator-valued function on the support of f, that is $\phi_a(x) = \phi_a(f(x))$, the third point of axiom Axiom W1 (Covariance) now reads as

- For transformations in the Poincaré group, which are given by the unitary representations $U(q, \Lambda, 0, 0)$, the field must satisfy

$$U(q, \Lambda, 0, 0)\phi_a(x)U(q, \Lambda, 0, 0)^\dagger = \phi_a(\Lambda x + q) \tag{2.58}$$

- For the special conformal transformation given by

$$x \mapsto x^b = \frac{x - \langle x, x\rangle b}{1 - 2\langle x, b\rangle + \langle x, x\rangle\langle b, b\rangle} \tag{2.59}$$

with unitary representation $U(0, 1, b, 0)$ the field must satisfy

$$U(0, 1, b, 0)\phi_a(x)U(0, 1, b, 0)^\dagger = N(x, b)^{-h_a}\phi_a(x^b) \tag{2.60}$$

where $N(x, b) = 1 - 2\langle x, b\rangle + \langle x, x\rangle\langle b, b\rangle$ and h_a is the **Conformal weight** of the field ϕ_a.

- For dilatations

$$x \mapsto x^\lambda = e^\lambda x \tag{2.61}$$

with unitary representation $U(0, 1, 0, \lambda)$ the field must satisfy

$$U(0, 1, 0, \lambda)\phi_a(x)U(0, 1, 0, \lambda)^\dagger = e^{\lambda d_a}\phi_a(x^\lambda) \tag{2.62}$$

with the **Scaling dimension** d_a, which is strictly related to the conformal weight just defined.

Similar definitions and extensions are defined for the 2 dimensional case but are not presented here.

Having defined CFTs axiomatically, we are now ready to analyze the operators in this theory, starting by looking at the action of infinitesimal conformal transformations on operators or fields.

Every conformal transformation can be represented as a differential operator acting on functions or fields. Given an infinitesimal transformation $x \to x' = x'(x)$ we can always write it in the form

$$x'^\mu = x^\mu + \omega_a \frac{\delta x^\mu}{\delta \omega_a} \tag{2.63}$$

where ω_a is small and $\frac{\delta x^\mu}{\delta \omega_a}$ indicates a variation in the coordinate with respect to a variation in ω_a. For example, in case of a translation, ω_a is actually ω^ν and $\frac{\delta x^\mu}{\delta \omega^\nu} = \delta^\mu_\nu$ so that $x'^\mu = x^\mu + \omega^\mu$ as expected.

Given any function $\phi(x)$, the generator G_a of the transformation is defined with the following relation

$$iG_a\phi(x) = \frac{\delta x^\mu}{\delta \omega_a}\partial_\mu\phi(x) \tag{2.64}$$

One can show (see [Qua15] for a complete reference), that

- Translations $x'^\mu = x^\mu + a^\mu$ are generated by $P_\mu = -i\partial_\mu$
- Rigid rotations $x'^\mu = M^\mu_\nu x^\nu$ are generated by $L_{\mu\nu} = i(x_\mu\partial_\nu - x_\nu\partial_\mu)$.
- Dilatations $x'^\mu = \alpha x^\mu$ are generated by $D = -ix^\mu\partial_\mu$.
- Special conformal transformations $x'^\mu = \frac{x^\mu - b^\mu\langle x,x\rangle}{1-\langle b,x\rangle+\langle b,b\rangle\langle x,x\rangle}$ are generated by $K_\mu = -i(2x_\mu x^\nu\partial_\nu - \langle x, x\rangle\partial_\mu)$.

Computing the commutators among these operators we obtain the conformal algebra:

$$\begin{aligned}
[D, P_\mu] = iP_\mu, \quad [D, K_\mu] = -iK_\mu, \quad [K_\mu, P_\nu] = 2i(g_{\mu\nu}D - L_{\mu\nu}), \\
[L_{\mu\nu}, P_\rho] = -i(g_{\mu\rho}P_\nu - g_{\nu\rho}P_\mu), \quad [L_{\mu\nu}, K_\rho] = -i(g_{\mu\rho}K_\nu - g_{\nu\rho}K_\mu), \\
[L_{\mu\nu}, L_{\rho\sigma}] = -i(L_{\mu\rho}g_{\nu\sigma} - L_{\mu\sigma}g_{\nu\rho} - L_{\nu\rho}g_{\mu\sigma} + L_{\nu\sigma}g_{\mu\rho}), \\
[D, L_{\mu\nu}] = 0, \quad [P_\mu, P_\nu] = 0, \quad [K_\mu, K_\nu] = 0, \quad [D, D] = 0
\end{aligned} \tag{2.65}$$

An important observation is $L_{\mu\nu}$ and P_μ form the Poincaré group, while it can be seen that $L_{\mu\nu}, P_\mu$ and D form a subgroup as well. Therefore, conformal symmetry is not implied by just Poincaré and dilatation transformations.

In $d = 2$ there also exists an infinite dimensional algebra of conformal transformations we will later study: the Virasoro algebra.

States, operators and quantization

We consider a multi-component operator $\phi_a(x)$ (there is more than one component if the operator has non-zero spin), where components are indexed by a. Using the above notation of infinitesimal generators, from the covariance axiom and the commutation relations (2.65) on translations, we can write

$$e^{-ixP}\phi_a(0)e^{ixP} = \phi_a(x) \tag{2.66}$$

where $xP = x^\mu P_\mu$. If we take the partial derivative with respect to x^μ we get

$$\partial_\mu\phi_a(x) = e^{-ixP}(-iP_\mu\phi_a(0) + \phi_a(0)iP_\mu)e^{ixP} = -i[P_\mu, \phi_a(x)] \tag{2.67}$$

where in the last equality we used the fact that P_μ and e^{ixP} commute. Hence, we deduce the action of the generator P_μ on the field operator

$$[P_\mu, \phi_a(x)] = i\partial_\mu\phi_a(x) \tag{2.68}$$

We declare the following actions on operators at the origin $\phi_a(0)$

$$\begin{aligned}
[D, \phi_a(0)] &= i\Delta\phi_a(0) \\
[L_{\mu\nu}, \phi_a(0)] &= i(S_{\mu\nu})^b_a\phi_b(0) \\
[K_\mu, \phi_a(0)] &= 0
\end{aligned} \tag{2.69}$$

where Δ is the scaling dimension and $S_{\mu\nu}$ is a matrix that depends on the spin of the field (if ϕ_a is multi-component) and is 0 for scalar fields.

These relations are very important since they define the primary operators

Definition 2.20. A **primary operator** of scaling dimension Δ is an operator ϕ_a for which the equations (2.69) are satisfied. Derivatives of primary operators are called **descendants**.

Given these, one can derive the general relations for $\phi_a(x)$:

$$\begin{aligned}
[P_\mu, \phi_a(x)] &= i\partial_\mu\phi_a(x) \\
[D, \phi_a(x)] &= i(\Delta + x^\mu\partial_\mu)\phi_a(x) \\
[L_{\mu\nu}, \phi_a(x)] &= -i(x_\mu\partial_\nu - x_\nu\partial_\mu)\phi_a(x) - i(S_{\mu\nu})^b_a\phi_b(x) \\
[K_\mu, \phi_a(x)] &= 2ix_\mu\Delta\phi_a(x) + i(2x_\mu x^\nu\partial_\nu - x^2\partial_\mu)\phi_a(x) + 2ix^\rho(S_{\rho\mu})^b_a\phi_b(x)
\end{aligned} \tag{2.70}$$

where, as usual, $x_\mu = g_{\mu\nu}x^\nu$ and $x^2 = g_{\mu\nu}x^\mu x^\nu = x_\mu x^\mu$.

Radial quantization

Recall that for Quantum Field Theories the natural foliation was

$$L(t_0) = \{x \in \mathbb{R}^{1,3} : x^0 = t_0\} \simeq \mathbb{R}^3 \tag{2.71}$$

and each time slice had its own Hilbert space of states. In general, the best choice for a foliation is given by the symmetries of the theory we are considering. In QFT time foliation works very well with Poincaré invariance and the evolution of leaves is in the P_0 operator.

Consider now a Conformal Field Theory in the Euclidean space $\mathbb{R}^d$. In this case, it is more convenient to foliate the space by spheres $R \cdot \mathbb{S}^{d-1}$ with origin at the center. Actually, we can center spheres around any point, provided that they are equivalent. In this case, we may want to correlate operators inserted inside the sphere with others inserted outside. This kind of slicing is called **radial quantization**. To move from surface to surface we just need to apply the dilatation operator with its representation $U = e^{iD\Delta\tau}$ with $\tau = \log r$ and r is the distance between the two radii.

In QFT, states are usually identified by their momentum which, recalling our discussion on QM, is given by the operators P_μ so that $P_\mu |k\rangle = k_\mu |k\rangle$. This correspondence is well defined since P_0 commutes with every P_μ and we can characterize well the correspondence between states on different time slices.

In CFT, it is common to classify operators according to the scale dimension and their spin using the eigenvalues:

$$\begin{aligned} D |\Delta, l\rangle &= i\Delta |\Delta, l\rangle \\ L_{\mu\nu} |\Delta, l\rangle_\alpha &= i(S_{\mu\nu})^\beta_\alpha |\Delta, l\rangle_\beta \end{aligned} \tag{2.72}$$

Scalars will be often denoted simply as $|\Delta\rangle$, using only their scaling dimension.

We will now analyze how insertions of operators work, starting from the vacuum state that we now call $|0\rangle$. The vacuum corresponds to no insertion at all and has a 0 eigenvalue for dilatations and 0 spin. Recall that Axiom W4 (Uniqueness of the Vacuum) means that the vacuum is invariant under conformal transformations.

Starting from $|0\rangle$ we can insert a primary operator at the origin $\phi_\Delta(0)$ and obtain a state $|\Delta\rangle = \phi_\Delta(0) |0\rangle$ which has indeed dilatation eigenvalue Δ:

$$D |\Delta\rangle = D\phi_\Delta(0) |0\rangle = [D, \phi_\Delta(0)] |0\rangle + \phi_\Delta(0) |0\rangle = i\Delta\phi_\Delta(0) |0\rangle = i\Delta |\Delta\rangle \tag{2.73}$$

We consider this as a primary state in the sense that the actions of the operators P, K, L are similar to the ones for primary operators in (2.69). Actually, the action of such operators on the state comes from the inserted operator, as we can see here:

$$K |\Delta\rangle = K\phi_\Delta(0) |0\rangle = [K, \phi_\Delta(0)] |\Delta\rangle = 0 \tag{2.74}$$

where we have used $K |0\rangle = 0$.

For now, we have inserted an operator at the origin. If we insert an operator in a generic point x, such as $|\psi\rangle = \phi_\Delta(x) |0\rangle$, we can see that this is not an eigenstate of the dilatation operator:

$$\begin{aligned} |\psi\rangle = \phi_\Delta(x) |0\rangle &= e^{-ixP}\phi_\Delta(0)e^{ixP} = e^{-ixP}\phi_\Delta(0) |0\rangle \\ &= |\Delta\rangle - ix \cdot P |\Delta\rangle - \frac{1}{2}(x \cdot P)^2 |\Delta\rangle + \ldots = \sum_{n\geq 0} \frac{1}{n!}(ix \cdot P)^n |\Delta\rangle \end{aligned} \tag{2.75}$$

Notice that the eigenvalue of the state $P_\mu |\Delta\rangle$ under dilatations is $\Delta + 1$:

$$DP_\mu |\Delta\rangle = ([D, P_\mu] + P_\mu D) |\Delta\rangle = i(\Delta + 1)P_\mu |\Delta\rangle \tag{2.76}$$

since $[D, P_\mu] = iP_\mu$. However, since P_μ is a derivative, the new state $P_\mu |\Delta\rangle$ is no more a primary:

$$K_\nu P_\mu |\Delta\rangle = ([K_\nu, P_\mu] + P_\mu K_\nu)\Delta = [K_\nu, P_\mu]\Delta \neq 0 \tag{2.77}$$

But this also means that $|\psi\rangle$ can be written a linear combination of a primary $|\Delta\rangle$ and all its descendants! This will be fundamental for the Operator Product Expansion in CFT.

As seen above, P_μ acting on a state $|\Delta\rangle$ increases its scaling dimension by one unit. This process can be repeated, obtaining

$$|\Delta\rangle \xrightarrow{P_\mu} |\Delta + 1\rangle \xrightarrow{P_\nu} |\Delta + 2\rangle \tag{2.78}$$

On the other hand, the operator K_μ actually lowers the dimension by one since

$$DK_\mu |\Delta + n\rangle = ([D, K_\mu] + K_\mu D) |\Delta + n\rangle = i(\Delta + n - 1)K_\mu |\Delta + n\rangle \tag{2.79}$$

since $[D, K_\mu] = -iK_\mu$ and, similarly, we have for example

$$0 \xleftarrow{K_\mu} |\Delta\rangle \xleftarrow{K_\nu} |\Delta + 1\rangle \xleftarrow{K_\rho} |\Delta + 2\rangle \tag{2.80}$$

This is important since it also gives a formal reason for the introduction of primary operators. Assuming that the dimensions are bounded from below, applying repetitively K_μ eventually yields 0, which means we have obtained a primary operator. This process shows us that starting from the insertion of a primary operator of energy Δ at the origin we obtain a state which gets annihilated by K_μ.

From a state-operator correspondence point of view, we have a procedure to obtain the primary operator of some states. Given a state with dilatation eigenvalue Δ which gets annihilated by K_μ, we can construct a local primary operator of dimension Δ. For such an operator, the correlator is given by

$$\langle \phi(x_1)\phi(x_2)\cdots\phi_\Delta(0)\rangle = \langle 0|\phi(x_1)\phi(x_2)\cdots|\Delta\rangle \tag{2.81}$$

We are now ready to study the action and the correlators for CFTs and, as we will shortly see, conformal symmetry unlocks many powerful tools compared to standard QFTs.

2.4 Correlators and the Operator Product Expansion

We will now study the fundamental quantities in QFT, that is correlators, in this new and more powerful setting of Conformal Field Theory. We start by looking at how correlators transform. Then, we will see how to exploit conformal symmetry to calculate 2, 3 and 4-point functions.

Recall that the action of a Lagrangian $\mathcal{L}$ is defined as

$$S[\phi] = \int \mathrm{d}x^d \mathcal{L}(\phi, \partial_\mu \phi) \tag{2.82}$$

where the Lagrangian is a function of a collection of fields and their derivatives. Consider a conformal transformation $x \mapsto x' = x'(x)$ and the field $\phi'(x')$ such that $\phi'(x') = \phi(x)$. From the above relations, one can compute that

$$\phi'(x') = \left|\frac{\partial x'}{\partial x}\right|^{-\frac{\Delta}{d}} \phi(x) \tag{2.83}$$

and since the Jacobian of the transformation is a function of the conformal factor

$$\left|\frac{\partial x'}{\partial x}\right| = \Lambda(x)^{-\frac{d}{2}} \tag{2.84}$$

we finally have

$$\phi'(x') = \Lambda(x)^{\frac{\Delta}{2}} \phi(x) \tag{2.85}$$

Defining the following transformation

$$\begin{aligned} x &\mapsto x' = x'(x) \\ \phi(x) &\mapsto \phi'(x') = \mathcal{F}(\phi(x)) \end{aligned} \tag{2.86}$$

we can calculate the transformed action

$$\begin{aligned} S[\phi'] &= \int \mathrm{d}x^d \mathcal{L}(\phi'(x), \partial_\mu \phi'(x)) \\ &= \int \mathrm{d}x'^d \mathcal{L}(\phi'(x'), \partial'_\mu \phi'(x')) \\ &= \int \mathrm{d}x'^d \mathcal{L}(\mathcal{F}(\phi(x)), \partial'_\mu \mathcal{F}(\phi(x))) \\ &= \int \mathrm{d}x^d \left|\frac{\partial x'}{\partial x}\right| \mathcal{L}\left(\mathcal{F}(\phi(x)), \frac{\partial x^\mu}{\partial x'^\nu} \partial_\nu \mathcal{F}(\phi(x))\right) \end{aligned} \tag{2.87}$$

We are now going to consider only actions that are invariant under conformal transformations, meaning that our theory has conformal symmetry. In particular, we will assume that both the measure $[d\phi]$ and the action $S[\phi]$ are invariant under conformal transformations. Also, from now on we assume to be working with scalar field operators so that $\phi(x)$ can be considered as a smooth function.

Consider now a general correlator and apply a transformation $x \mapsto x'$. Since the relations for descendants can be obtained by differentiating the ones on primaries, we can work with only primary operators.

$$\begin{aligned} \int [d\phi]\phi(x'_1)\cdots\phi(x'_n)e^{-S[\phi]} &= \int [d\phi']\phi'(x'_1)\cdots\phi'(x'_n)e^{-S[\phi']} \\ &= \int [d\phi]\mathcal{F}(\phi(x_1))\cdots\mathcal{F}(\phi(x_n))e^{-S[\phi]} \end{aligned} \tag{2.88}$$

where in the first equality we simply renamed the integration variable and in the second we applied the transformation, remembering that the measure and the action are both invariant. For the Poincaré group, we have that

$$\begin{aligned} &\mathcal{F}(\phi(x)) = \phi'(x+a) = \phi(x) \;\; a \in \mathbb{R}^d \\ &\mathcal{F}(\phi(x)) = \phi'(\Lambda x) = \phi(x) \end{aligned} \tag{2.89}$$

since translations and rigid rotation have unit determinants. Therefore we can conclude

$$\begin{aligned}\langle\phi(x_1+a)\cdots\phi(x_n+a)\rangle &= \langle\phi(x_1)\cdots\phi(x_n)\rangle \\ \langle\phi(\Lambda x_1)\cdots\phi(\Lambda x_n)\rangle &= \langle\phi(x_1)\cdots\phi(x_n)\rangle\end{aligned} \tag{2.90}$$

On the other hand, for a general conformal transformation, the determinant is no more one and the fields transform as (2.83). For example, dilatations have a determinant of $\left|\frac{\partial x'}{\partial x}\right| = \lambda^d$ (or $e^{\lambda d}$, depending on the representation we are using). In general, the relation is given by

$$\langle\phi(x_1)\cdots\phi(x_n)\rangle = \left|\frac{\partial x'}{\partial x}\right|_{x=x_1}^{\frac{\Delta_1}{d}} \cdots \left|\frac{\partial x'}{\partial x}\right|_{x=x_n}^{\frac{\Delta_n}{d}} \langle\phi(x_1')\cdots\phi(x_n')\rangle \tag{2.91}$$

As a final remark, notice that these correlators involve the same fields inserted at different points in space.

Two-point functions of scalars

Let's see how two-point functions $\langle\phi_1(x_1)\phi_2(x_2)\rangle$ are fixed through conformal transformations. Poincaré invariance implies that

$$\begin{aligned}\langle\phi_1(x_1)\phi_2(x_2)\rangle &= \langle\phi_1(0)\phi_2(x_2-x_1)\rangle \\ &= \langle\phi_1(0)\phi_2(\Lambda(x_2-x_1))\rangle = \langle\phi_1(\Lambda(x_1-x_2))\phi_2(0)\rangle\end{aligned} \tag{2.92}$$

for any rigid rotation Λ. Two point functions are therefore radial $\langle\phi_1(x_1)\phi_2(x_2)\rangle = f(|x_1-x_2|)$ with f smooth. If we now apply a scale transformation $x \mapsto x' = \lambda x$ we get

$$\langle\phi_1(x_1)\phi_2(x_2)\rangle = \lambda^{\Delta_1+\Delta_2}\langle\phi_1(\lambda x_1)\phi_2(\lambda x_2)\rangle \tag{2.93}$$

from which $f(r) = \lambda^{\Delta_1+\Delta_2} f(\lambda r)$. Radial functions homogeneous with exponent α are of the form $\frac{C}{r^\alpha}$ and hence

$$\langle\phi_1(x_1)\phi_2(x_2)\rangle = \frac{C_{12}}{|x_1-x_2|^{\Delta_1+\Delta_2}} \tag{2.94}$$

Now we focus on special conformal transformations, which have determinant

$$\left|\frac{\partial x'}{\partial x}\right| = \frac{1}{(1-2b\cdot x+b^2x^2)^d} \tag{2.95}$$

where we have used the notation $a\cdot b = a^\mu b^\nu g_{\mu\nu}$ and $a^2 = a\cdot a$. With some calculations and using the definition of special conformal transformation we can show

$$|x_1'-x_2'| = \frac{|x_1-x_2|}{\gamma_1^{\frac{1}{2}}\gamma_2^{\frac{1}{2}}} \tag{2.96}$$

where $\gamma_i = 1-2b\cdot x_i + b^2x_i^2$. If we apply the transformation to the two-point function we have

$$\begin{aligned}\langle\phi_1(x_1)\phi_2(x_2)\rangle &= \frac{C_{12}}{|x_1-x_2|^{\Delta_1+\Delta_2}} \\ &= \frac{1}{\gamma_1^{\Delta_1}\gamma_2^{\Delta_2}}\frac{C_{12}}{|x_1'-x_2'|^{\Delta_1+\Delta_2}} \\ &= \frac{(\gamma_1\gamma_2)^{\frac{\Delta_1+\Delta_2}{2}}}{\gamma_1^{\Delta_1}\gamma_2^{\Delta_2}}\frac{C_{12}}{|x_1-x_2|^{\Delta_1+\Delta_2}}\end{aligned} \tag{2.97}$$

where in the second equality we applied the special conformal transformation on the correlator and, in the third, we used (2.96). Since γ_1 and γ_2 are independent this can be satisfied only if $\Delta_1 = \Delta_2$. We can finally conclude

Theorem 2.9. *Two primary scalar fields are correlated only if they have the same scaling dimension*

$$\langle \phi_1(x_1)\phi_2(x_2)\rangle = \begin{cases} 0 & \Delta_1 \neq \Delta_2 \\ \frac{C_{12}}{|x_1 - x_2|^{2\Delta}} & \Delta_1 = \Delta_2 = \Delta \end{cases} \tag{2.98}$$

We now introduce the notation $x_{12} = x_1 - x_2$ to shorten the subsequent relations.

Three point functions of scalars

Following the steps in [Qua15], in a Poincaré invariant theory the three-point function is of the form

$$\langle \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\rangle = \frac{C_{123}}{|x_{12}|^a |x_{23}|^b |x_{13}|^c} \tag{2.99}$$

From scale invariance we get

$$\frac{C_{123}}{|x_{12}|^a |x_{23}|^b |x_{13}|^c} = \frac{\lambda^{\Delta_1+\Delta_2+\Delta_3}}{\lambda^{a+b+c}} \frac{C_{123}}{|x_{12}|^a |x_{23}|^b |x_{13}|^c} \tag{2.100}$$

hence

$$a + b + c = \Delta_1 + \Delta_2 + \Delta_3 \tag{2.101}$$

Using special conformal transformation we obtain

$$\frac{C_{123}}{|x_{12}|^a |x_{23}|^b |x_{13}|^c} = \frac{(\gamma_1\gamma_2)^{\frac{a}{2}} (\gamma_2\gamma_3)^{\frac{b}{2}} (\gamma_1\gamma_3)^{\frac{c}{2}}}{\gamma_1^{\Delta_1} \gamma_2^{\Delta_2} \gamma_3^{\Delta_3}} \frac{C_{123}}{|x_{12}|^a |x_{23}|^b |x_{13}|^c} \tag{2.102}$$

Since we want again this to be true for every x_1, x_2, x_3 and the γ_i are independent, we get the final system

$$\begin{aligned} a + b + c &= \Delta_1 + \Delta_2 + \Delta_3 \\ a + c &= 2\Delta_1 \\ a + b &= 2\Delta_2 \\ b + c &= 2\Delta_3 \end{aligned} \tag{2.103}$$

from which

$$\begin{aligned} a &= \Delta_1 + \Delta_2 - \Delta_3 \\ b &= \Delta_2 + \Delta_3 - \Delta_1 \\ c &= \Delta_3 + \Delta_1 - \Delta_2 \end{aligned} \tag{2.104}$$

Finally, we have obtained the following:

Theorem 2.10. *Three primary scalar fields are correlated with the following relation*

$$\langle \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\rangle = \frac{C_{123}}{|x_{12}|^{\Delta_1+\Delta_2-\Delta_3} |x_{23}|^{\Delta_2+\Delta_3-\Delta_1} |x_{13}|^{\Delta_3+\Delta_1-\Delta_2}} \tag{2.105}$$

Since we can normalize states and operators, C_{ij} and C_{ijk} do not both contain relevant physical information. In particular, we could normalize one of them and the value of the other gets locked by our choice. The most common choice is to put $C_{ij} = 1$. This means that the non-trivial physical information is contained in the value of C_{ijk} and the latter will depend on the specific theory we are studying.

In the end, both 2 and 3-point functions are written in terms of just the distance between the points of insertion of operators, the scaling dimensions and the non-trivial physical constants C_{ijk} known as structure constants or OPE coefficients.

Four-point functions

When dealing with four-point functions the situation is very different and we no more have a closed-form expression. We need to construct more complex objects and go back to the theory of states and operators.

As a start, define the following quantities, which do not exist for less than 4 points and are invariant to all conformal transformations

$$u = \frac{x_{12}^2 x_{34}^2}{x_{13}^2 x_{24}^2}, \quad v = \frac{x_{14}^2 x_{23}^2}{x_{13}^2 x_{24}^2} \tag{2.106}$$

These are called **cross-ratios**. It can be shown that the general form of a four-point function is

$$\langle \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\rangle = \frac{F(u,v)}{\prod_{i<j} |x_{ij}^2|^{\delta_{ij}}} \tag{2.107}$$

where $\sum_{j\neq i} \delta_{ij} = \Delta_i$. $F(u,v)$ could be any function of the cross-ratios, at least in theory. In fact, we will show that it has indeed a particular form. This same method can also be applied to correlators of more than 4 points. To obtain additional information on this kind of correlator, we need to take a step back to the Operator Product Expansion which, for CFT, is much more powerful.

Operator Product Expansion

Recall that, in QFT, the OPE let us write a product of two operators, inserted in two points close to each other, as an asymptotic expansion: an infinite sum that for each truncation converges to the same value as the original product, but just for a single point near to the points of insertion. This is a limitation since the expansion is therefore not convergent in general. In this case, conformal invariance together with the states-operator correspondence imply an actual convergence.

Consider the insertion of two operators inside a sphere and the generated state

$$|\psi\rangle = \phi_1(x)\phi_2(0)\,|0\rangle \tag{2.108}$$

Without loss of generality, we can assume one of them to be inserted at the origin. As for QM, we can expand the state $|\psi\rangle$ in a basis of eigenstates of the dilatation operator:

$$|\psi\rangle = \sum_{n\geq 1} c_n(x)\,|E_n\rangle \tag{2.109}$$

where c_n is in general a function of x. As shown before, each $|E_n\rangle$ can be expressed as a linear combination of primaries and their descendants yielding

$$\phi_1(x)\phi_2(0)\,|0\rangle = \sum_{\phi \text{ primaries}} D_\Delta(x,\partial)\phi_\Delta(0)\,|0\rangle \tag{2.110}$$

where $D_\Delta(x,\partial)$ is a power series of partial derivatives (which produce the descendant operators) depending on x.

Notice how the radial quantization, together with the concept of primary operators and descendants, leads to an expansion that, it can be shown, is actually convergent. For a proof of this fact we refer to [PRER12] and [Pol98] but, when dealing with a correlator

$$\langle \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\cdots\phi_n(x_n)\rangle, \tag{2.111}$$

the expansion converges for every $x_2 \in \mathbb{R}^d$ such that $|x_1 - x_2| < |x_1 - x_k|, k = 3,\dots,n$, that is if $\phi_2(x_2)$ is the operator inserted closest to x_1. We could also say that the radius of convergence is given by $\min\{|x_1 - x_k|, k = 3,\dots,n\}$.

Conformal invariance lets us also determine some properties of the terms $C_\Delta(x,\partial)$. For example (see [Ryc17] for a review), one can show that

$$\phi_1(x)\phi_2(0)\,|0\rangle = \frac{D}{|x|^k}[\phi_\Delta(0) + \dots]\,|0\rangle + \text{ contribution of other primaries} \tag{2.112}$$

where the exponent k is fixed by scaling invariance to $k = \Delta_1 + \Delta_2 - \Delta$. Expanding the first descendant term leads to

$$\phi_1(x)\phi_2(0)\,|0\rangle = \frac{D}{|x|^{\Delta_1+\Delta_2-\Delta}}(\phi_\Delta(0) + \alpha x^\mu \partial_\mu \phi_\Delta(0) + \dots)\,|0\rangle + \dots \tag{2.113}$$

and the value of α gets also fixed by conformal invariance to the value

$$\alpha = \frac{\Delta_1 - \Delta_2 + \Delta}{2\Delta} \tag{2.114}$$

In fact, we could go ahead and determine all the coefficients! It has been proven that conformal invariance fixes completely the functions $D_\Delta(x,\partial)$ up to an overall factor $C_{12\Delta}$, which is the same structure constant in 2.105 for three point functions. Note finally that $D_\Delta(x,\partial)$ depends solely on the three scaling dimensions $\Delta_1, \Delta_2, \Delta$.

We are now ready to go back to four-point functions. Consider four scalar primaries, assumed to be identical for simplicity, with scaling dimension Δ_E. As before, because of conformal invariance, we start by writing

$$\langle \phi(x_1)\phi(x_2)\phi(x_3)\phi(x_4)\rangle = \frac{G(u,v)}{|x_{12}|^{2\Delta_E}|x_{34}|^{2\Delta_E}} \tag{2.115}$$

Then, we apply the OPE to

$$\begin{aligned} \phi(x_1)\phi(x_2) &= \sum_\Delta C_\Delta D_\Delta(x_{12},\partial_y)\phi_\Delta(y)\big|_{y=\frac{x_1+x_2}{2}} \\ \phi(x_3)\phi(x_4) &= \sum_{\Delta'} C_{\Delta'} D_{\Delta'}(x_{34},\partial_z)\phi_{\Delta'}(z)\big|_{z=\frac{x_3+x_4}{2}} \end{aligned} \tag{2.116}$$

where the C_Δ are called **OPE coefficients** and are the same as the $C_{12\Delta}$ used above. Hence we can rewrite the correlator as

$$\langle \phi(x_1)\phi(x_2)\phi(x_3)\phi(x_4)\rangle = \sum_{\Delta,\Delta'} C_\Delta C_{\Delta'}[D_\Delta(x_{12},\partial_y)D_{\Delta'}(x_{34},\partial_z)\langle\phi_\Delta(y)\phi_{\Delta'}(z)\rangle] \tag{2.117}$$

Given the above discussion on two-point functions, we can simplify this equation further since the two-point correlator is not zero if and only if the two field operators have the same scaling dimension $\Delta = \Delta'$. We can therefore write

$$\langle \phi(x_1)\phi(x_2)\phi(x_3)\phi(x_4)\rangle = \sum_{\Delta} C_{\Delta}^2 [D_{\Delta}(x_{12}, \partial_y) C_{\Delta}(x_{34}, \partial_z)\langle \phi_{\Delta}(y)\phi_{\Delta}(z)\rangle] \tag{2.118}$$

where again $y = \frac{x_1+x_2}{2}$ and $z = \frac{x_3+x_4}{2}$. Now recall that both the functions $D_{\Delta}(x_{12}, \partial_y)$ and the two-point functions are fixed by conformal symmetry. This means that the whole object inside the brackets is fixed:

$$[D_{\Delta}(x_{12}, \partial_y) D_{\Delta}(x_{34}, \partial_z)\langle \phi_{\Delta}(y)\phi_{\Delta}(z)\rangle] = \frac{G_{\Delta,l}(u,v)}{|x_{12}|^{2\Delta_E}|x_{34}|^{2\Delta_E}} \tag{2.119}$$

where the functions $G_{\Delta,l}(u,v)$ are called **conformal blocks** which depend on the scaling dimensions Δ_E, Δ and the spin l we now reintroduce. Therefore the four-point function has the following expansion

$$G(u,v) = \sum_{\Delta,l} C_{\Delta,l}^2 G_{\Delta,l}(u,v) \tag{2.120}$$

The conformal blocks $G_{\Delta,l}(u,v)$ for 4 point functions of scalar operators are known functions in general dimension and, in particular, are the eigenfunctions of a second order differential operator called the **Casimir operator**. See [DO04] and [DO01] for a reference.

As a final remark, notice how this process can be applied to higher-order correlators. In conformal field theories, n-point functions can be written solely in terms of the dimension of primaries, OPE coefficients and the structure of the Operator Product Expansion itself.

2.4.1 Conformal invariance in the Euclidean plane

We now have a quick look at the case of the Euclidean plane $\mathbb{R}^{2,0}$. Recall that in 2 dimension the algebra of conformal transformations is infinite-dimensional. This, as we will shortly see, can be used to exploit additional information and completely solve some theories.

As before, we identify the euclidean plane $\mathbb{R}^{2,0}$ with the complex plane $\mathbb{C}$ with the conformal transformations corresponding exactly to the holomorphic and anti-holomorphic functions. Consider as notations for the complex derivatives

$$\begin{aligned} \partial = \partial_z &= \frac{1}{2}(\partial_0 - i\partial_1) \\ \bar{\partial} = \partial_{\bar{z}} &= \frac{1}{2}(\partial_0 + i\partial_1) \end{aligned} \tag{2.121}$$

We introduce the following generators for transformations

Definition 2.21. The generators for the holomorphic and anti-holomorphic transformations are given by

$$l_n = -z^{n+1}\partial_z, \quad \bar{l}_n = -\bar{z}^{n+1}\partial_{\bar{z}} \tag{2.122}$$

These generators have the algebra given by

$$\begin{aligned}[l_m, l_n] &= (m-n)l_{m+n} \\ [\bar{l}_m, \bar{l}_n] &= (m-n)\bar{l}_{m+n} \\ [l_m, \bar{l}_n] &= 0\end{aligned} \tag{2.123}$$

The conformal algebra is then a direct sum of two infinite dimensional algebras, called Witt algebras. Unfortunately, not all the generators are well defined on the compactification of $\mathbb{C}$, the whole Riemann sphere $S^2 = \mathbb{C} \cup \infty$. The only ones to be globally defined are

$$\{l_{-1}, l_0, l_1\} \cup \{\bar{l}_{-1}, \bar{l}_0, \bar{l}_1\} \tag{2.124}$$

which form the global conformal algebra. We can recover the usual conformal transformations discussed in previous chapters by identifying

- $l_{-1}, \bar{l}_{-1}$ as the generators for translations.
- $l_0 + \bar{l}_0$ as the generator of dilatations.
- $i(l_0 - \bar{l}_0)$ as the generator of rotations.
- $l_1, \bar{l}_1$ as generators of special conformal transformations.

It is also convenient to work with a basis of eigenstates of l_0 and $\bar{l}_0$ with eigenvalues h and $\bar{h}$, called the **conformal weights** of the state. The usual scaling dimension Δ and spin s of a state are then given by

$$\Delta = \frac{h + \bar{h}}{2}, \quad s = \frac{h - \bar{h}}{2}. \tag{2.125}$$

Fields in two-dimensional CFTs can be expressed as functions of $z, \bar{z}$ and written as $\phi(z, \bar{z})$. We now introduce a categorization of fields similar to the one adopted for general CFT.

Definition 2.22. A conformal field $\phi(z)$ is called a **primary field** of conformal weights $(h, \bar{h})$ if under conformal transformations $z \mapsto z' = f(z)$ transforms as

$$\phi'(z', \bar{z}') = \left(\frac{\partial f}{\partial z}\right)^{-h} \left(\frac{\partial \bar{f}}{\partial \bar{z}}\right)^{-\bar{h}} \phi(z, \bar{z}) \tag{2.126}$$

A conformal field with this property being satisfied only for global conformal transformations generated by (2.124) is called **quasi-primary**.

A general n-point correlator of n primary fields of conformal weights $(h_i, \bar{h}_i)$ transforms as

$$\langle \phi_1(z_1', \bar{z}_1') \cdots \phi_n(z_n', \bar{z}_n') \rangle = \prod_{i=1}^{n} \left(\frac{\partial f}{\partial z}\right)^{-h_i}_{z=z_i} \left(\frac{\partial \bar{f}}{\partial \bar{z}}\right)^{-\bar{h}_i}_{z=z_i} \langle \phi_1(z_1, \bar{z}_1) \cdots \phi_n(z_n, \bar{z}_n) \rangle \tag{2.127}$$

As before, two-point and three-point functions are fixed by global conformal invariance.

For two-point functions, the correlator is non-null if and only if $h_1 = h_2 = h$ and $\bar{h}_1 = \bar{h}_2 = \bar{h}$, giving the relation

$$\langle \phi_1(z_1, \bar{z}_1) \phi_2(z_2, \bar{z}_2) \rangle = \frac{C_{12}}{z_{12}^{2h} \bar{z}_{12}^{2\bar{h}}} \tag{2.128}$$

Three-point functions are indeed given by

$$\begin{aligned}&\langle\phi_1(z_1,\bar{z}_1)\phi_2(z_2,\bar{z}_2)\phi_3(z_3,\bar{z}_3)\rangle = \\ &= \frac{C_{123}}{z_{12}^{h_1+h_2-h_3}z_{23}^{h_2+h_3-h_1}z_{13}^{h_1+h_3-h_2}\bar{z}_{12}^{\bar{h}_1+\bar{h}_2-\bar{h}_3}\bar{z}_{23}^{\bar{h}_2+\bar{h}_3-\bar{h}_1}\bar{z}_{13}^{\bar{h}_1+\bar{h}_3-\bar{h}_2}}\end{aligned} \tag{2.129}$$

As before, we aim to determine the four-point function. In this case, we only have one relevant cross-ratio given by

$$\eta = \frac{z_{12}z_{34}}{z_{13}z_{24}} \tag{2.130}$$

The fact that we only need one cross-ratio is explained by the fact that conformal transformations can map any four points of the Riemann sphere (z_1, z_2, z_3, z_4) into $(0, 1, \eta, \infty)$. This also gives an intuitive idea of how two and three-point functions are completely fixed.

Without going in-depth with the actual quantization applied on the Euclidean plane CFT we define a new algebra of operators similar to (2.123), called the Virasoro algebra

Definition 2.23. The **Virasoro algebra** with **central charge** $(c, \bar{c})$ is given by

$$\begin{aligned}[L_n, L_m] &= (n-m)L_{n+m} + \frac{c}{12}n(n^2-1)\delta_{n+m,0} \\ [L_n, \bar{L}_m] &= 0 \\ [\bar{L}_n, \bar{L}_m] &= (n-m)\bar{L}_{n+m} + \frac{\bar{c}}{12}n(n^2-1)\delta_{n+m,0}\end{aligned} \tag{2.131}$$

Note how the commutation relations for $L_{\pm1}$, L_0 and the anti-holomorphic counterparts do not depend on the central charge and agree with the relations for $l_0, l_{\pm1}$ within the Witt algebra, generating the global conformal transformations. One can show that the state obtained by the insertion of a primary $\phi_{h,\bar{h}}(0,0)$ at the origin satisfies

$$\begin{aligned}&L_0\left|h,\bar{h}\right\rangle = h\left|h,\bar{h}\right\rangle, \quad &&\bar{L}_0\left|h,\bar{h}\right\rangle = \bar{h}\left|h,\bar{h}\right\rangle, \\ &L_n\left|h,\bar{h}\right\rangle = 0, &&\bar{L}_n\left|h,\bar{h}\right\rangle = 0 \quad \text{for } n>0\end{aligned} \tag{2.132}$$

We have not put any restriction on the possible values for the central charge c and the values of the conformal weights $(h, \bar{h})$. An important physical constraint we can impose is that states should always have a positive norm. This is often referred as **unitarity** and is very important for the probabilistic interpretation of QM. Unitarity also imposes that the squared OPE coefficients are real and positive [RRTV08]. Given a primary state $|h\rangle$, consider $L_{-n}|h\rangle$.

$$\begin{aligned}\langle h|L_nL_{-n}|h\rangle &= \langle h|\left(L_{-n}L_n + 2nL_0 + \frac{1}{12}cn(n^2-1)\right)|h\rangle \\ &= \left(2nh + \frac{1}{12}cn(n^2-1)\right)\langle h|h\rangle\end{aligned} \tag{2.133}$$

Letting n be arbitrarily large we obtain $c > 0$ as a requirement for unitarity, while from using $n = 1$ we get $h \geq 0$. Further considerations can add more restrictions on possible values of the central charge and conformal weights. In particular, having $c > 1$ and $h > 0$ always ensures to have unitarity, while for $0 < c < 1$ we have that

- The central charge has to be of the form

$$c(m) = 1 - \frac{6}{m(m+1)} \quad m = 2, 3, 4, \ldots \tag{2.134}$$

- For each m there is only a finite number of primaries leading to a unitary representation. Such primaries have dimension

$$h_{r,s}(m) = \frac{((m+1)r - ms)^2 - 1}{4m(m+1)}, \tag{2.135}$$

  where $1 \leq r \leq m-1$ and $1 \leq s \leq r$.

These quantities define the so-called **minimal models** which were completely solved using just the constraints provided by Virasoro symmetry [BPZ84]. For CFTs in more than 2 dimensions, we need to take advantage of other symmetries and ideas to gain additional information on the scaling dimensions and OPE coefficients.

2.5 Conformal Bootstrap

What we have obtained in the previous section is an expansion of the four-point function in terms of the squared OPE coefficients and the conformal blocks, written as a function of the cross ratios u, v. Another important fact is that the latter can be expressed as functions of new variables $z, \bar{z}$ which, in the Euclidean CFT case, are conjugate values in the complex plane, while in Lorentzian signature they obey to $z, \bar{z} \in (0, 1)$.

$$u = z\bar{z} = \frac{x_{12}^2 x_{34}^2}{x_{13}^2 x_{24}^2}, \quad v = (1-z)(1-\bar{z}) = \frac{x_{14}^2 x_{23}^2}{x_{13}^2 x_{24}^2} \tag{2.136}$$

or, equivalently,

$$z, \bar{z} = \frac{1}{2} = \left(u - v + 1 \pm \sqrt{(u-v+1)^2 - 4u}\right). \tag{2.137}$$

With these new coordinates the operator product expansion still converges exponentially in Δ [PRER12], [Pol98] for all complex points $z, \bar{z}$ except for the half-lines on the real axis $(-\infty, 0]$ and $[1, \infty)$.

However, in the previous expansion, we have chosen to expand the fields inserted at specific points by coupling $\phi(x_1)$ with $\phi(x_2)$ and $\phi(x_3)$ with $\phi(x_4)$. This is sometimes referred as the **s-channel expansion**.

This choice is indeed arbitrary, although as we will see not all the pairings are useful. If we instead decide to expand, using the OPE, the products $\phi(x_1)\phi(x_4)$ and $\phi(x_2)\phi(x_3)$, the final correlator has to be the same function! Looking closely, this new way of expanding the four-point function is just the equivalent of switching the cross rations $u \leftrightarrow v$ or the points $x_1 \leftrightarrow x_3$. This is often called as the **t-channel expansion**. Since the four-point correlator must not change, the two four-point expansions must yield the same result

$$\frac{G(u,v)}{x_{12}^{2\Delta_\phi} x_{34}^{2\Delta_\phi}} = \frac{G(v,u)}{x_{23}^{2\Delta_\phi} x_{14}^{2\Delta_\phi}} \tag{2.138}$$

which leads to

$$v^{\Delta_\phi}\left(\sum_{\Delta,l} C^2_{\Delta,l} G_{\Delta,l}(u,v)\right) = u^{\Delta_\phi}\left(\sum_{\Delta,l} C^2_{\Delta,l} G_{\Delta,l}(v,u)\right) \tag{2.139}$$

Therefore, we have shown that the conformal blocks and the OPE coefficients must respect the following

$$\sum_{\Delta,l} C^2_{\Delta,l}\left(v^{\Delta_\phi} G_{\Delta,l}(u,v) - u^{\Delta_\phi} G_{\Delta,l}(v,u)\right) = 0 \tag{2.140}$$

which is referred as the **Conformal bootstrap equation**. This is a fundamental result in CFT since imposing this equation for any u, v or $z, \bar{z}$ we have an additional constraint on the OPE coefficient and the spectrum of the operators.

An interesting question is what happens if we use the last possibility of coupling together $\phi(x_1)$ with $\phi(x_3)$ and $\phi(x_2)$ with $\phi(x_4)$. One could impose the above equality for this channel as well, although it can be proven that there is no further information on the coefficients and conformal blocks that can be obtained from this.

2.5.1 Historical approaches and success

Initial studies on the representations of the conformal group and the Operator Product Expansion in the CFT framework, where it has a finite radius of convergence, can be found in [FGG73]. Soon thereafter, the crossing equation has been studied by Polyakov [Pol74] as a way to extract additional and useful equations without the Lagrangian formalism. This is indeed a fundamental result for the theory since often the Lagrangian is unknown or the coupling constant is too big to rely on perturbation theory, one of the few approaches available for computations in QFT.

This approach lead to a successful and complete classification of the so-called **minimal models** by Polyakov, Belavin and Zamolodchikov in [BPZ84] as we discussed above. The main idea is to exploit the additional symmetries given by the Virasoro algebra in 2 dimensions obtaining particular restrictions on theories with a central charge $0 < c < 1$.

Given their complexity, the first attempts to bootstrap higher dimensional CFTs only came in 2008 with the work [RRTV08], where it was shown that CFTs must have a scalar operator with scaling dimension $\Delta_{\min} \leq f(d)$ where $f(d)$ is a function of the space-time dimension with $f(1) = 2$. If we write the conformal bootstrap equation in the form

$$\sum_{\Delta,l} C^2_{\Delta,l} F_{d,\Delta,l}(z,\bar{z}) = 0, \tag{2.141}$$

the unitarity condition guarantees that the coefficients $C^2_{\Delta,l}$ are real and positive [RRTV08], which is crucial for the following approach. Focusing on the line $z = \bar{z}$ with $0 \leq z \leq 1$ for even values of the spin l, one can study numerically the conformal bootstrap equation to put further constraints on the admissible values of the scaling dimensions Δ and the OPE coefficients. Further results and reviews of the studies on these topics can be found in [SD16, PRV19], but the main procedure can be stated as follows:

- Make a hypothesis on the scaling dimensions and spins Δ, l appearing in the Operator Product Expansion of our interests.

- Search for a non-negative linear functional α acting on the functions $F_{d,\Delta,l}$, being strictly positive on at least one of them. A good choice is to consider the action of the functional as

$$\alpha(F) = \sum_{m+n\leq\Lambda} \alpha_{mn}\partial_z^n\partial_{\bar{z}}^m F(z,\bar{z})|_{z=\bar{z}=\frac{1}{2}} \tag{2.142}$$

  where the particular point $z = \bar{z} = \frac{1}{2}$ is chosen for convergence reasons.

- If such α exists, then the hypothesis is wrong by applying the functional to both sides of equation (2.141), recalling that α is strictly positive for at least one of the conformal blocks.

The main problem remains to find the functional α and the fact that we are dealing with continuous values for Δ and the OPE coefficients.

Possible solutions are to take only discrete values of the scaling dimensions up to a cutoff Δ_{max} or to approximate the derivatives of the conformal blocks with polynomials. To run the search for the functional the two main approaches are semidefinite programming [SD15] or the navigator bootstrap presented in [RRSD+21].

One very interesting result within this framework comes from [KPSD14], where a particular set of allowed 3 dimensional CFTs was restricted in a small island around the $3D$ Ising model, possibly claiming that this is the only theory allowed in this region of the plane in figure 2.1

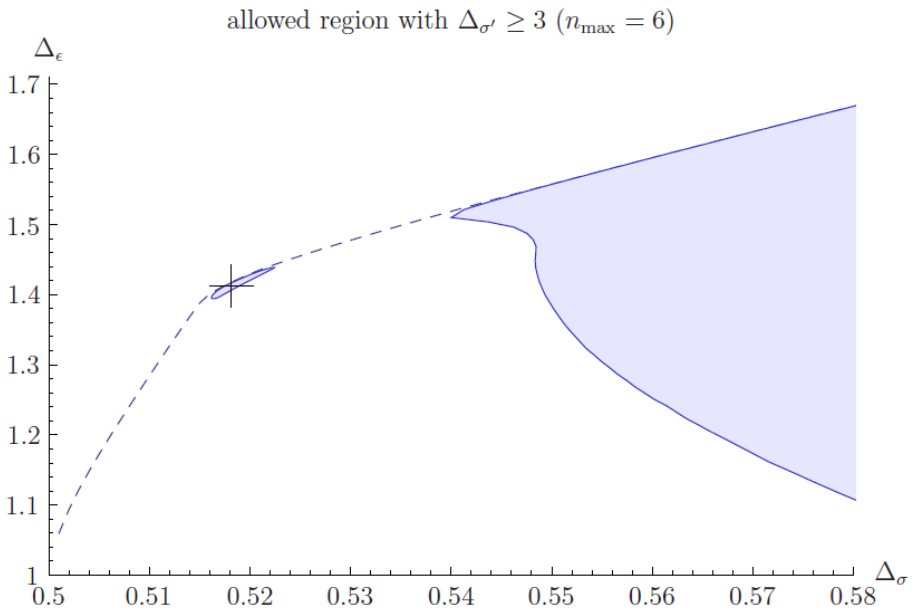


Figure 2.1: Allowed region for a particular set of 3 dimensional CFTs, taken from [KPSD14]

The above methods are mostly numerical, studying approximating values of the conformal blocks to exclude regions for allowed theories.

Recently, researchers began to develop alternative approaches to the traditional conformal bootstrap such as Montecarlo [LVS22] and Machine Learning methods. In this thesis we focus on the particular application of Reinforcement Learning to CFTs as in [KPN22a, KPN22b, KNPR23].

Chapter 3

Reinforcement Learning Generalities

The increased computational power of modern computers and the technological progress in the past decades led to increased interest and research in Machine Learning (ML). ML techniques found application in a large variety of topics due to their flexibility and adaptability. Neural Networks, for example, can approximate up to arbitrary precision any function from the interval $[0, 1]$ to itself [HSW89, Cyb89]. Any real problem that can be reformulated into a regression task of this kind could be solved using an ML algorithm.

This power comes with consequences, as Neural Networks are known for the very hard optimization problem of finding the optimal weights, as well as the risk of constructing a model that performs very well on the training set without being able to generalize on unseen data. Therefore, ML often requires finding the sweet spot between a complex model, which can learn perfectly the data we feed, and a simpler one, able to perform better on the test set.

Reinforcement Learning is a particular ML technique involving two components: an agent and the environment. At each step, the agent performs an action and receives from the environment a reward, as well as some kind of information. RL algorithms aim to find the optimal strategy for the agent that can maximize the expected cumulative reward from a whole interaction. Although a greedy solution may be tempting, in some problems we could have early choices leading to a greater initial reward but worst results in the long run.

RL techniques often involve creating a model representing the agent-environment interaction or finding a link between the actions taken, the information and the rewards obtained. As we will see, Neural Networks fit well for this particular task.

In this chapter we study the main concepts around ML and RL, starting from the basic training algorithms and building up towards the algorithm of our interest: Soft Actor-Critic.

3.1 Machine Learning

In this first section, we introduce the main concepts around Machine Learning following the discussion of [GBC16]. We start with the most general definitions related to all ML algorithms, followed by an in-depth discussion on Feedforward Neural Networks with their optimization and training techniques.

3.1.1 Motivations and main concepts

Machine Learning is a very wide realm containing many different topics and concepts. A good summary of the most general idea of ML is the definition by Tom Mitchell [Mit97]:

Definition 3.1. An algorithm or computer program is said to **learn** from experience E with respect to some class of task T and performance measure P, if its performance at task T, as measured by P, improves with experience E.

We will soon give a clear example of how this definition can be interpreted in mathematical terms.

Recently, ML algorithms grew in diffusion and found large usage in many different tasks including translation, summarization and question answering (Natural Language Processing [OMK19]), autonomous driving (Computer Vision [YLCT20]), protein folding prediction (biochemistry [JEP+21]), as well as medicine and games. Common tasks of ML can mainly be divided into two categories:

- **Supervised learning**: supervised learning usually involves data with an input x and an output y. The algorithm learns a function $\hat{f}(x)$ which, with respect to the performance measure, must be as close as possible to the true target $f(x) = y$.
- **Unsupervised learning**: in unsupervised learning we are given only input data x. The algorithm learns the properties of given data or even how to generate new synthetic samples with the same properties. For example, if we suppose that the given samples x are generated by an underlying distribution $\mathbb{P}(x)$, unsupervised learning is used to distinguish real samples generated by $\mathbb{P}(x)$ from false examples from another distribution. Furthermore, we can use the input data x to also generate new synthetic samples coming from a learned distribution as close as possible to the original one.

Example 3.2. Although being solved much before the introduction of ML, a very simple and common example of an ML task is **linear regression**. Linear regression is useful both for a complete understanding of definition 3.1 and because it constitutes the basis for more complex models such as neural networks. We are given some input data points $x^{(i)} \in \mathbb{R}^n$ and each one of them has an output value $y^{(i)} \in \mathbb{R}$, for $i = 1, 2, \dots, m$ where m is the number of available samples. We construct the set of linear functions $y = w^T x + b$ with parameters $w \in \mathbb{R}^n, b \in \mathbb{R}$. We aim to find the best parameters with respect to the mean squared error between the true outputs $y^{(i)} = f(x^{(i)})$ and our linear estimate $\hat{y}^{(i)} = w^T x^{(i)} + b$. Our minimization problem has then the form

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \|\hat{y}^{(i)} - y^{(i)}\|_2^2. \tag{3.1}$$

This problem has an actual explicit solution: for b the best value corresponds to the mean of the true outputs y_i. We can assume without loss of generality that the outputs have 0 mean. We write the data matrix as $X \in \mathbb{R}^{m \times n}$, where the i-th row of the matrix is the i-th input data point. We finally have

$$w = (X^T X)^{-1} X^T Y, \tag{3.2}$$

where Y is the column vector of outputs $Y = (y^{(1)}, \dots, y^{(m)})$.

However, our algorithm should not only perform accurately on known data but it should also be able to generalize well on unseen samples, as we could just memorize the data we trained it on. To model this situation, we suppose that the **training set**, the set of samples used to train the algorithm, as well as the **validation** and **test sets**, which are used to simulate the performance of the model on unseen data, are generated from a distribution over datasets p_{data} with the following assumptions:

- For each dataset all samples are independent from each other.
- All the datasets are generated from the same distribution p_{data}.

Assume that the input and output data are related through a function with noise $y = f(x) + \epsilon$. We will consider ϵ to have a normal distribution $\mathcal{N}(0, \sigma^2)$ and to be independent of the data samples and the data distribution. The ML model consists of a set of functions, the **hypothesis space** $\mathcal{H}$, and an algorithm that has to select the best estimator $\hat{f}$ from the hypothesis space, based on the available data. In the case of a regression task, one of the most common error metrics is the mean squared error $\mathbb{E}\left[(y - \hat{f}(x))^2\right]$, where the expectation is taken with respect to the data distribution and ϵ.

From a probabilistic and statistical point of view, a point estimate $\hat{f}(x)$ is a random variable and a function of training data D, $\hat{f}(x, D)$, although for simplicity we will keep the notation $\hat{f}(x)$. The choice of this estimate is dependent on the training data, which is supposed to be generated through p_{data}. As a consequence, we can calculate the mean, the variance and other statistical quantities for $\hat{f}(x)$ with the data distribution $\mathbb{E}_D\left[\hat{f}(x)\right]$.

The mean squared error $\mathbb{E}_{D,\epsilon}\left[(f(x) - \hat{f}(x))^2\right]$, with $x \sim D$, decomposes as

$$\begin{aligned} MSE &= \mathbb{E}_{D,\epsilon,x}\left[(y - \hat{f}(x))^2\right] \\ &= \sigma^2 + \mathbb{E}_x\left[(f(x) - \mathbb{E}_D[\hat{f}(x)])^2\right] + \mathbb{E}_{D,x}\left[(\mathbb{E}_D[\hat{f}(x)] - \hat{f}(x))^2\right] \\ &= \sigma^2 + \text{Bias}(\hat{f}) + \text{Variance}(\hat{f}) \end{aligned} \tag{3.3}$$

Hence, the error is made of 3 components:

- The variance of noise σ^2, also called standard error: this is a kind of error we cannot avoid and sets a lower bound for how precise our model can be.
- The bias term $\mathbb{E}_{D,x}\left[(f(x) - \mathbb{E}_D[\hat{f}])^2\right]$: it is the error between the true value and the average of our point estimation. It can be regarded as an error due to our assumptions about the problem and the hypothesis class we are considering. If we are limited to the use of a restricted set of functions, this term increases. This is often related to the concept of **underfitting**: the model lacks enough complexity to estimate the objective function.
- The variance term $\mathbb{E}_{D,x}\left[(\mathbb{E}_D[\hat{f}] - \hat{f}(x))^2\right]$: it is indeed the variance of our estimator being a random variable and function of the data. It tells us how much our point estimate varies with different training sets and experiences. High variance is often related to **overfitting**, where the estimator adapts too well to the training set without being able to generalize well.

In reality, since the data distribution p_{data} is unknown, the **empirical risk** is used in practical applications as an estimator of the mean squared error

$$\frac{1}{N} \sum_{\text{training set}} \|y - \hat{y}\|_2^2, \tag{3.4}$$

which assumes equal weights for all independent samples in the training set.

3.1.2 Deep Feedforward Networks

Deep Feedforward Networks, also called **Feedforward Neural Networks** or **Multilayer Perceptrons** are the fundamental models in modern deep learning. Neural Networks represent a function $f(x)$, with the term feedforward used to indicate that the computations needed to calculate f are done in a sequential forward manner, without any feedback to previous operations. The name "Networks", on the other hand, comes from the different intermediate functions that compose $f(x) = f^{(k)}(f^{(k-1)}(\dots(f^{(1)}(x))))$ and from their visual representation.

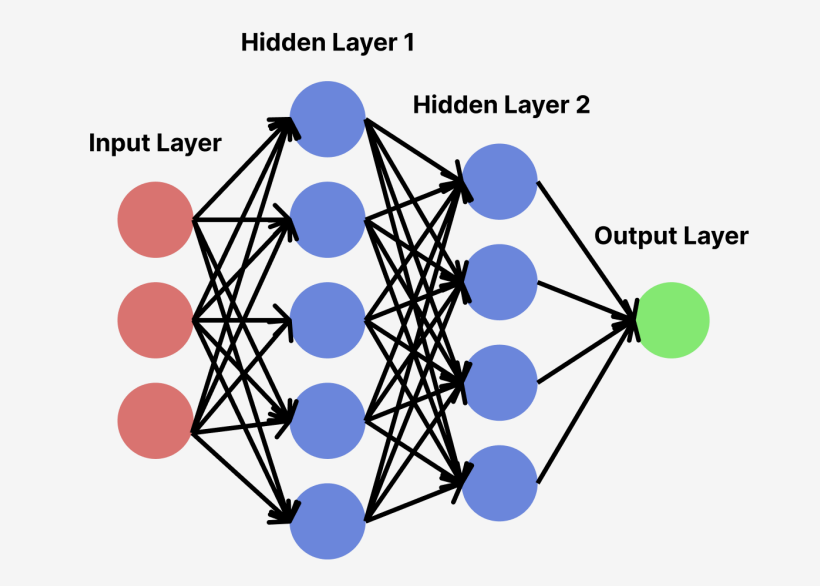


Figure 3.1: Graphical representation of a Feedforward Neural Network. The circles represent the units while the arrows represent the connections. The direction of the arrows indicates the direction of the calculations and the flow of information during inference. Data flows from the input layer (red) through the hidden layers (blue) and finally the output layer (green). Note that there are no connections between units of the same layer.

The intermediate functions $f^{(i)}$ are called the **layers** of the network and the whole length of the chain of computation is the **depth**. The first layer is usually called the input layer, the intermediate ones are hidden layers and the last one is the output layer. Each layer is actually constituted by **units**, also called **neurons**, which act parallel to each other and perform the same kind of computation.

Each unit performs a scalar product between the input values it receives from all neurons from the previous layer and a set of weights w, a non-linear function is then applied to obtain the final output or **activation** of the unit. We now analyze in depth all the procedures and the calculations of the output of a neural network.

Input layer

The input layer is the first component of the neural network and it is composed of a number of neurons equal to the dimension of the input. There are no calculations at this level, the units are placeholders for the input data. Therefore we indicate the units of the input layer with the components of the input vector $(x_1, x_2, \dots, x_n)$.

Hidden layers

In hidden layers, each unit receives as input the activations of neurons from the previous layer and calculates its activation by a scalar product and a non-linear function. Mathematically, let k_l be the number of units for layer l and let the activations of neurons from layer $l-1$ be indicated as $(a_1^{l-1}, \dots, a_{k_{l-1}}^{l-1})$. Let the weights of unit i in layer l be $w_{i1}^l, w_{i2}^l, \dots, w_{ik_{l-1}}^l$. As equivalent notations, we will write the vector of activations from the previous layer as $\boldsymbol{a}^{l-1} = (a_1^{l-1}, \dots, a_{k_{l-1}}^{l-1})$, the weights of unit i in layer l as $\boldsymbol{w}_i^l = (w_1^l, \dots, w_{k_{l-1}}^l)$ and the entire set of weights of layer l with $\boldsymbol{W}^l$ where W_{ij}^l is the weight of unit i for the activation a_j^{l-1}. Let also $\boldsymbol{b}^l = (b_1^l, \dots, b_{k_l}^l)$ be the biases of neurons of layer l. The activation of unit i is then $a_i^l = g(\boldsymbol{w}_i^l \cdot \boldsymbol{a}^{l-1} + b_i^l)$ which is a linear combination of previous activations followed by a function g which is non-linear. We can also write the activations of layer l as $\boldsymbol{a}^l = g(\boldsymbol{W}^l \boldsymbol{a}^l + \boldsymbol{b}^l)$ where g is applied element-wise in this notation.

The activation function g plays a fundamental role since it adds a non-linear component to the network. If we choose not to use it, the output of layer l can be written as $\boldsymbol{a}^l = \boldsymbol{W}^l \boldsymbol{W}^{l-1} \boldsymbol{W}^{l-2} \cdots \boldsymbol{W}^1 \boldsymbol{x} = \boldsymbol{W}_{\text{equiv}} \boldsymbol{x}$ where we have suppressed the biases for simplicity. This means that if no activation is used then the neural network is a linear regressor and its output is a linear combination of the inputs.

The choice of the activation function is really important since it determines the non-linearity of the network. Common choices at initial stages of neural network research were the hyperbolic tangent function $\tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$ and the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Although they are bounded and differentiable everywhere, they present an issue often called **saturation**. Since these functions have finite limits for $x \to \pm\infty$, all inputs sufficiently distant from 0 produce very similar outputs and, during the training phase, gradients close to 0.

A huge improvement was made with the introduction of the ReLU activation function [Fuk69, NH10]. The ReLU function is defined as $g(x) = \max\{0, x\}$ and brings three main advantages:

- It induces sparsity since all negative inputs are mapped into 0, acting as a regularization.
- It is differentiable almost everywhere and the derivative is easy to compute. It avoids the introduction of second-order effects in gradient which increases the speed of learning.
- For positive inputs there is no plateau effect and the gradient does not approach zero.

On the other hand, the ReLU function is not differentiable at $x = 0$ and we could argue that mapping any negative input into 0 could hugely affect the performance of the model by losing information. Although true, these two issues seem to have a minor

effect compared to the actual advantages it brings and the ReLU function is the most used activation function for hidden layers.

Output layer

The output layer is the final step of the calculation and the value we obtain is the output of the neural network. The number of units corresponds to the dimension of the space where our target values live, at least for regression tasks. The calculations for each unit are the same as for the hidden layer neurons although, in this case, it is sometimes better to use limited activation functions. The sigmoid function, for example, is often used in case we want outputs with values in $(0, 1)$, such as for probabilities and images.

When dealing with classification tasks, on the other hand, we want to find the correct label among d values. In these cases, the output layer is constituted by d units, each representing the probability of the input being in the corresponding class. For this kind of task, the most common activation function for the output layer is the softmax and it has a particularity: instead of taking as input a single real value from another, it operates on vectors, ensuring as well that all the components sum up to 1.

$$\text{softmax}(x_1, x_2, \ldots, x_d) = \left(\frac{e^{x_1}}{\sum_i e^{x_i}}, \frac{e^{x_2}}{\sum_i e^{x_i}}, \ldots, \frac{e^{x_d}}{\sum_i e^{x_i}} \right) \tag{3.5}$$

One of the biggest problems with Feedforward Neural Networks is deciding how many layers it should have and how many units should each layer contain, which defines the hypothesis space for this model. Deeper networks are often able to generalize well even with fewer units but require many subsequent computations, especially for the training procedure. On the other hand, adding too many parameters can lead to overfitting with the model practically memorizing the training samples. A common procedure to find the best parameters is model selection. While training the model with a set of parameters on the training set, the validation set is unbiased to validate and simulate its performance on unseen data. We select the parameters that bring the best accuracy and the final model is selected and tested on the test set.

3.1.3 Training

When building a neural network, the initial weights and biases of the units are chosen randomly since finding the optimal values in advance is a complex and costly process. Instead, the main paradigm of ML consists in selecting randomly the weights and adjusting them as we gain information and experience with the data we feed the model. This translates into a complex optimization problem which is also not independent from the initialization of parameters as we will see later. Luckily there are ways to mitigate those problems although the main task of the training procedure is the weight adjustment.

As described above, when the neural network receives an input x the information flows in the network in a forward way through all of its layers, producing the output estimate $\hat{y}$. This procedure is called forward propagation and, during the training phase, it produces a scalar cost as a function of the current parameters $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$. In general, the choice of the cost function is very important and depends on the objective and the outputs:

- For regression tasks where a real output value is produced, usual choices are the mean squared error (MSE) $\frac{1}{N}\sum_{\text{training set}} \|y - \hat{y}\|_2^2$ or the mean absolute error (MAE) $\frac{1}{N}\sum_{\text{training set}} \|y - \hat{y}\|_1$.

- For classification tasks where we need to identify a label from C categories for each input, a good choice is the log-likelihood. The network outputs the probabilities of being in each category and the loss for a single training sample is calculated as $\sum_{c=1}^{C} \delta_{y=c} \log p_c(x)$ where $p_c(x)$ is the output probability of being in category c.

Backpropagation

In order to optimize the parameters of the Feedforward Neural Network we will use a gradient method: given the training loss, we calculate the gradient of the cost function with respect to the parameters and modify the weights and biases to reduce the error, following the gradient. Computing the gradient analytically is not complex but computing it for each unit individually has a huge computational cost as the number of units increases. Therefore, we need a way to compute gradients in a fast and reliable manner.

The main algorithm for gradient computation in neural networks is **backpropagation** [RHW86]. Its name is inspired by the idea of information on the error flowing back from the output layer, where the empirical risk is calculated, to the individual weights in the units of previous layers. Backpropagation is a very general algorithm and applies not only to feedforward networks but also to more complex ML models.

To fully understand it, we start by computing the gradient of the cost function with respect to a single weight in a unit. Consider a multilayer neural network with L total layers and layer l has k_l units. As above, the activation of unit i in layer l is given by $a_i^l = g(\boldsymbol{w}_i^l \cdot \boldsymbol{a}^{l-1} + b_i^l)$ where g is any activation function of our choice with the only requirement of it being differentiable. For simplicity, we call the quantity $\boldsymbol{w}_i^l \cdot \boldsymbol{a}^{l-1} + b_i^l$ as z_i^l, the preactivation of unit i in layer l. Recall w_{ij}^l is the weight of neuron i in layer l applied to the activation of neuron j in the previous layer and b_i^l is the bias of neuron i in layer l. The relations in matrix form are $\boldsymbol{a}^{l+1} = g(\boldsymbol{W}^{l+1}\boldsymbol{a}^l + \boldsymbol{b}^{l+1})$ where $W_{ij}^l = w_{ij}^l$ and g is applied component by component on vectors.

Consider the case of a network producing outputs $\hat{y} \in \mathbb{R}^d$, actually being the activations of the last layer $\hat{y} = \boldsymbol{a}^L = o(\boldsymbol{w}^L \cdot \boldsymbol{a}^{L-1} + b^L)$ where o is the activation function of the output layer, although we will consider with abuse of notation $g = o$. To have the most general setting possible, we consider a general loss function $L(y, \hat{y})$ between the true target and the output of the network. Since the cost function is the average of single errors on samples, we just need to compute it analytically on a single sample. We start by taking the partial derivative vector of the loss with respect to the

activations of layer l, $\boldsymbol{a}^l$.

$$\begin{aligned}
\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^l} &= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \cdot \frac{\partial \boldsymbol{a}^L}{\partial \boldsymbol{a}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} \frac{\partial \boldsymbol{a}^{h+1}}{\partial \boldsymbol{a}^h} \cdot \frac{\partial \boldsymbol{a}^{l+1}}{\partial \boldsymbol{a}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} \frac{\partial \boldsymbol{a}^{h+1}}{\partial \boldsymbol{z}^{h+1}} \frac{\partial \boldsymbol{z}^{h+1}}{\partial \boldsymbol{a}^h} \cdot \frac{\partial \boldsymbol{a}^{l+1}}{\partial \boldsymbol{z}^{l+1}} \frac{\partial \boldsymbol{z}^{l+1}}{\partial \boldsymbol{a}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} (g')|_{\boldsymbol{z}^{h+1}} \boldsymbol{W}^{h+1} \cdot (g')|_{\boldsymbol{z}^{l+1}} \boldsymbol{W}^{l+1}
\end{aligned} \tag{3.6}$$

where $(g')|_{\boldsymbol{z}^{h+1}}$ is the diagonal matrix with elements $(g')_{ii}|_{\boldsymbol{z}^{h+1}} = (g')\left(z_i^{h+1}\right)$, while the quantities $\frac{\partial \boldsymbol{a}^{h+1}}{\partial \boldsymbol{a}^h}$ indicate the jacobian matrices. This is indeed a row vector as the first element on the left of the product is a row vector, while all the other elements are matrices. Note that the information flows left to right from the last layer to the previous layers, just like the order of row vector-matrix multiplication.

We are now ready to differentiate with respect to the actual weights and bias of unit i in layer l obtaining

$$\begin{aligned}
\frac{\partial L(y,\hat{y})}{\partial w_{ij}^l} &= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \cdot \frac{\partial \boldsymbol{a}^L}{\partial \boldsymbol{a}^l} \frac{\partial \boldsymbol{a}^l}{\partial w_{ij}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} (g')|_{\boldsymbol{z}^{h+1}} \boldsymbol{W}^{h+1} \cdot (g')|_{\boldsymbol{z}^{l+1}} \boldsymbol{W}^{l+1} \cdot \frac{\partial \boldsymbol{a}^l}{\partial \boldsymbol{z}^l} \frac{\partial \boldsymbol{z}^l}{\partial w_{ij}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} (g')|_{\boldsymbol{z}^{h+1}} \boldsymbol{W}^{h+1} \cdot (g')|_{\boldsymbol{z}^{l+1}} \boldsymbol{W}^{l+1} \cdot (g')_{\cdot i}|_{\boldsymbol{z}^l} \cdot a_j^{l-1} \\
\frac{\partial L(y,\hat{y})}{\partial b_i^l} &= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \cdot \frac{\partial \boldsymbol{a}^L}{\partial \boldsymbol{a}^l} \frac{\partial \boldsymbol{a}^l}{\partial b_i^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} (g')|_{\boldsymbol{z}^{h+1}} \boldsymbol{W}^{h+1} \cdot (g')|_{\boldsymbol{z}^{l+1}} \boldsymbol{W}^{l+1} \cdot \frac{\partial \boldsymbol{a}^l}{\partial \boldsymbol{z}^l} \frac{\partial \boldsymbol{z}^l}{\partial b_{ij}^l} \\
&= \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \prod_{h=l+1}^{L-1} (g')|_{\boldsymbol{z}^{h+1}} \boldsymbol{W}^{h+1} \cdot (g')|_{\boldsymbol{z}^{l+1}} \boldsymbol{W}^{l+1} \cdot (g')_{\cdot i}|_{\boldsymbol{z}^l}
\end{aligned} \tag{3.7}$$

where $(g')_{\cdot i}|_{\boldsymbol{z}^l}$ indicates the i-th column vector of the diagonal matrix, which consists of $(0, \ldots, (g')|_{z_i^l}, \ldots, 0)^T$. Those quantities are indeed real values and are the components of the gradient with respect to single weights and biases of units. To better see the recursive procedure of backpropagation consider the gradient of the loss function with respect to activations of layer l as

$$\begin{aligned}
\delta^l &= \nabla_{\boldsymbol{a}^l} L(y,\hat{y}) = \frac{\partial L(y,\hat{y})}{\partial \boldsymbol{a}^L} \cdot \frac{\partial \boldsymbol{a}^L}{\partial \boldsymbol{a}^l} \\
&= \nabla_{\boldsymbol{a}^l} \boldsymbol{a}^L \nabla_{\boldsymbol{a}^L} L(y,\hat{y}) \\
&= \left(\boldsymbol{W}^{l+1}\right)^T \cdot (g')|_{\boldsymbol{z}^{l+1}} \cdot \prod_{h=l+1}^{L-1} \left(\boldsymbol{W}^{h+1}\right)^T (g')|_{\boldsymbol{z}^{h+1}} \nabla_{\boldsymbol{a}^L} L(y,\hat{y}) \\
&= \left(\boldsymbol{W}^{l+1}\right)^T \cdot (g')|_{\boldsymbol{z}^{l+1}} \delta^{l+1}
\end{aligned} \tag{3.8}$$

with $\delta^L = \nabla_{\boldsymbol{a}^L} L(y, \hat{y})$. As we can see the computation of gradients in hidden layers has a very convenient recursive form. Furthermore, the fact that the recursion has a backward procedure in which the gradients of layer l can be computed from subsequent layers justifies the name of the process.

Backpropagation lets us save calculations from computing each partial derivative separately when propagating the error through the network. Gradients for layer l can be obtained by multiplying gradients from layer $l+1$ with the $\boldsymbol{W}^{l+1}$ weight matrix and the derivative of the activation function, applied on the preactivations of layer $l+1$.

The full training step of a feedforward neural network can then be summarized as in algorithm 1:

Algorithm 1 Training step for the backpropagation algorithm

Initialize parameters $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$.
for each input-output couple (x, y) **do**
  Compute the neural network output $\hat{y}$.
  Save all the intermediate preactivations and activations of the hidden layers $\boldsymbol{z}^l, \boldsymbol{a}^l$.
  Calculate the error with respect to the true target $L(y, \hat{y})$.
  Compute the gradient of the error w.r.t. the output $\nabla_{\hat{y}} L(y, \hat{y}) = \nabla_{\boldsymbol{a}^L} L(y.\hat{y})$.
  for each each layer l from $L-1$ to 1 **do**
    Compute the gradient of the loss w.r.t. the activation with the formula

$$\delta^l = \left(\boldsymbol{W}^{l+1}\right)^T \cdot (g')|_{\boldsymbol{z}^{l+1}} \, \delta^{l+1} \tag{3.9}$$

    with base case $\delta^L = \nabla_{\boldsymbol{a}^L} L(y, \hat{y})$.
    Compute the gradients w.r.t. the weights and biases of the single units as

$$\begin{aligned} \frac{\partial L(y, \hat{y})}{\partial w_{ij}^l} &= (g')_{\cdot i}|_{\boldsymbol{z}^l} \, a_j^{l-1} \cdot \delta^l \\ \frac{\partial L(y, \hat{y})}{\partial b_i^l} &= (g')_{\cdot i}|_{\boldsymbol{z}^l} \cdot \delta^l \end{aligned} \tag{3.10}$$

  end for
end for

3.1.4 Optimization

There is a fundamental difference between optimization theory and ML theory: the former aims to minimize a cost function $J(\boldsymbol{\theta})$ directly, while the latter optimizes a performance measure while operating on a cost function, which may be different. Mathematically, we can think of the true cost as the average loss with respect to the true data generating distribution, also called **risk** $J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}}[L(f(\boldsymbol{x};\boldsymbol{\theta}), y)]$, while the function we optimize is the average loss on the finite training set, that is the empirical risk $J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$. If we knew the true data-generating distribution, the risk minimization problem would become an optimization problem. Instead, we can only optimize the empirical risk on the training set, hoping that it will reduce the true objective as well.

Another important aspect of ML is that updating the parameters using gradient descent can be very costly: the loss is an average of the errors on each sample and, if

the training set is large, calculating the gradient and backpropagating it through the network takes time and resources. Estimating the gradient update from just a subset of the training samples is not only faster but may also solve the problem of redundant or unbalanced training sets. These methods are called **batch** methods if updates are calculated on the whole training set and **minibatch** methods if we only use a subset of it. While it may seem obvious that batch methods are more accurate in estimating the true gradient, minibatch learning can also act as a regularizer by adding noise to the gradient, while also being much faster in convergence [WM04].

One important aspect for minibatch learning to be effective is that the subsets of examples we calculate the gradient at each step (called **minibatches**) are independent of each other and their data samples are selected randomly and independently. If these conditions are met, by sampling a minibatch of data $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ we obtain an unbiased estimator of the exact gradient for the generalization error as

$$\hat{\boldsymbol{g}} = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) \tag{3.11}$$

Neural network optimization provides a variety of challenges to be faced and is indeed an extremely difficult task:

- Even if the objective function is convex, second-order effects caused by large eigenvalues in the Hessian matrix of the cost function may lead to poor optimization.

- Neural networks are, for the most part, non-convex problems to optimize and they suffer from a usual problem for non-convex functions: local minimums and flat regions. In practice, neural network models present a huge number of local minima. Nevertheless, there is a tendency for large networks to have even more saddle points and plateaus in addition to the local minimum [DPG+14] with the landscape of the cost as a function of the weights and biases of the neural network being complex [SMG14].

- Neural networks with a large number of hidden layers involve many multiplications of weights, which may produce very steep or very flat regions in the landscape of the cost function. Solutions for these problems include gradient clipping, which controls the maximum magnitude of the gradient.

- Minibatch methods produce a gradient estimate which is unbiased but can still be inexact producing wrong update directions.

- Network initialization plays a really important role: if the initial parameters are in a region of the landscape far from the optimal ones it can be challenging to move them away by using gradients that push parameters only towards local minima near the initialization.

Neural network optimization is a flourishing area of research due to its challenges and importance but theoretical results are still hard to find and prove due to all the problems and the large number of parameters that modern neural networks present.

Algorithms for neural network optimization

In order to face the many challenges of neural network optimization a series of algorithms have been developed. Since we are dealing with a differentiable function, the most common ideas involve gradient descent. The most used algorithms in ML, particularly for deep learning, are a modification of gradient descent called **Stochastic Gradient Descent (SGD)** and its variants [RM51], [BCN18]. SGD is a gradient descent based on minibatch learning: at each step, we sample m data points from the training set and average their gradient of the loss function. This produces an unbiased estimator of the true gradient which will be used as the direction for parameter update. A really important feature of SGD is that the scale of the step, also called **learning rate**, is not a fixed value ϵ but varies with time ϵ_k, where k is the time step. In particular, it is common to take a decreasing learning rate $\epsilon_{k+1} < \epsilon_k$. The main reason behind a varying learning rate is that SGD introduces noise by sampling a small subset of the training samples and this effect never vanishes, even when the true gradient is approximately 0 at a local minima.

To ensure convergence of the SGD it is sufficient to use a schedule of the learning rate such that:

$$\sum_{k=1}^{\infty} \epsilon_k = \infty, \qquad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty \tag{3.12}$$

Many theoretical results have been proven for the convergence of stochastic gradient descent under such a hypothesis. In particular, depending on the convexity of the cost and estimating function we have convergence in expectation to a local or global minimum (convex case) [BCN18], [RM51], [Bot91]. The learning rate regulates the speed of convergence: in some cases, if the initial learning rate is too small, there can be no learning at all, leaving the model stuck with a high-cost function. A common choice for the learning rate schedule is to take $\epsilon_k \sim \frac{1}{\sqrt{k}}$.

To further accelerate learning, many variations of stochastic gradient descent have been discussed in the literature and are now commonly used when training large neural networks. When dealing with noisy or small gradients a good choice is to update the parameters with a momentum, either in the Polyak variant [Pol64] or in its Nesterov variant [Nes83]. The momentum helps the parameters keep moving in the direction of past gradients, reducing the effects of noise in the last step and following the main direction of improvement. To increase efficiency even more and solve the problem of hyperparameter selection, adaptive learning rate algorithms have been introduced such as AdaGrad [DHS11] and RMSProp, both of which try to limit the effect of gradients accumulating and progressively increasing in value.

The most important and most used algorithm for neural network optimization is **Adam** [KB15] and is a combination of the previously seen methods of momentum and RMSProp, with some particularities. Adam includes both first-order and second-order exponentially decaying momentums and adds a bias correction on those terms. Adam is now widely used in neural network training and is the algorithm we will use. The pseudocode for the Adam algorithm is presented in algorithm 2.

Algorithm 2 Pseudocode for the Adam optimization algorithm

Initialize step size ϵ, decay rates for moment estimates ρ_1, ρ_2, numerical stability constant $\delta = 10^{-8}$.
Initialize parameters $\boldsymbol{\theta}$, first and second moment variables $\boldsymbol{s}, \boldsymbol{r}$, initial time step $t = 0$.
while stopping criterion not met **do**
  Sample a minibatch of m samples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with targets $\{y^{(i)}\}$.
  Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$.
  $t \leftarrow t + 1$
  Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$.
  Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$.
  Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$.
  Correct bias in second moment $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$.
  Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$.
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
end while

Usually $\epsilon \in [0.0001, 0.001]$ while $\rho_1, \rho_2 \in [0, 1)$, with defaults set respectively at 0.9 and 0.999.

3.2 Reinforcement Learning

Reinforcement Learning (RL) is a different paradigm of modern ML which involves two components: the agent and the environment. The agent has to learn how to pick actions in the environment to maximize a numerical reward signal. To do so, the agent explores the environment with a trial-and-error strategy looking for the best long-term strategy, or **policy**. Actions in the early stages of the interactions affect not only immediate rewards but also subsequent actions and reward signals.

Some of the most powerful algorithms for RL involve neural networks and other ML models to evaluate the state of the interaction between the agent or the environment or to select the next action. The data to train those models is continuously obtained through the exploration of the environment.

We now give a mathematical formalization of all the concepts and components of Reinforcement Learning by introducing the finite Markov decision processes. We will be following the book [SB18].

3.2.1 Finite Markov Decision Processes

Markov decision processes (MDP) best represent this interactive framework between the agent and the environment. We consider the two main components:

- **Agent**: it is the explorer, decision maker and learner of the RL framework.
- **Environment**: it represents everything outside the agent and gets explored by interacting with it.

We now define the main concepts of RL.

Definition 3.3. A **state** $S \in \mathcal{S}$ is a representation of the current situation of the environment. $\mathcal{S}$ is the set of all possible states.

Definition 3.4. An **action** $A \in \mathcal{A}(S)$ is a possible behavior of the agent and it depends on the current state. $\mathcal{A}(S)$ is the set of all possible actions given the current state S.

Definition 3.5. An **observation** $O \in \mathcal{O}$ is the response from the environment to the agent after an action. $\mathcal{O}$ is the set of all observations.

Definition 3.6. A **reward** $R \in \mathcal{R} \subset \mathbb{R}$ is a numerical signal the agent receives after an action and represents how well the agent performed with the last action.

In the case of a finite MDP the sets $\mathcal{A}, \mathcal{O}, \mathcal{S}, \mathcal{R}$ are all finite.

We then consider a sequence of discrete time steps, usually the integer line $0, 1, 2, \ldots$. At each time step t the agent has an internal state $S_t \in \mathcal{S}$ and decides an action $A_t \in \mathcal{A}(S_t)$ based on the current state and its understanding of the environment. At step $t+1$ the agent receives a reward $R_{t+1} \in \mathcal{R}$ and an observation $O_{t+1} \in \mathcal{O}$. Given these, the agent produces a new current state S_{t+1} and the process gets repeated iteratively.

Definition 3.7. A **trajectory** is a possibly infinite sequence of elements obtained by the process above in the form

$$S_0, A_0, R_1, O_1, S_1, A_1, R_2, O_2, S_2, A_2, \ldots \tag{3.13}$$

Definition 3.8. We call the **history** up to time t the current trajectory up to time t: $H_t = S_0, A_0, R_1, O_1, S_1, A_1, \ldots R_t, O_t$.

In general, the internal state of the agent at time t is a function of the history up to time t and it represents the current belief about the environment by the agent. It is the main information, coupled with the experience and strategy, that the agent uses to select the next action. In this sense, the state is a function of the history $S_t = f(H_t)$.

Since the sets are all finite the following distributions are well-defined:

Definition 3.9. The **transition probability distribution** $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the conditional probability for the new state given the past state and the action of the agent:

$$p(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a] \tag{3.14}$$

Definition 3.10. The **dynamics distribution** $p : \mathcal{R} \times \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ of the MDP is the joint probability distribution of the reward and next state given by the past state and the action taken by the agent:

$$p_d(r, s'|s, a) = \mathbb{P}[R_{t+1}, S_{t+1} = s'|S_t = s, A_t = a] \tag{3.15}$$

It is clear that $p(s'|s, a) = \sum_{r \in \mathcal{R}} p_d(s', r|s, a)$.

With an abuse of notation, we will indicate both functions with p. It will be clear from time to time which one is being used since the number of arguments is different.

Before giving the formal definition of a MDP we recall some important concepts for probability distribution related to Markov processes and stochastic processes

Definition 3.11. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A **filtration** $\mathcal{F}_t$ with $t \in \mathbb{R}$ is a set of sub-σ-algebras of $\mathcal{F}$ satisfying the property

$$\mathcal{F}_k \subseteq \mathcal{F}_l \quad \text{if} \quad k \leq l \tag{3.16}$$

Definition 3.12. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with a filtration $\mathcal{F}_t$. Let $(\Omega', \mathcal{G})$ a measurable space. A Ω' valued stochastic process $X = \{X_t : \Omega \to \Omega'\}_{t \in \mathbb{R}}$ adapted to the filtration (that is X_t is $\mathcal{F}'$ measurable) satisfies the **Markov property** if for each $F \in \mathcal{G}$ and each $s, t \in \mathbb{R}$ with $s < t$

$$\mathbb{P}(X_t \in F | \mathcal{F}_s) = P(X_t \in F | X_s) \tag{3.17}$$

If Ω' is discrete with the discrete σ-algebra and $t \in \mathbb{N}$, we can reformulate the property as

$$\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}) \tag{3.18}$$

Intuitively, for the discrete case, the last observation of the stochastic process contains all the available information at the time of the next step, it is indeed sufficient. The process has no memory, meaning that adding past observations does not add information.

All the above quantities and definitions are the fundamental concepts for a Markov decision process and give a mathematical and formal representation of the reinforcement learning framework we described: given the action taken, the internal state for the agent satisfies the Markov property. We are ready to give the formal definition of an MDP:

Definition 3.13. A **Markov Decision Process** is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p_d)$, where p_d is a probability distribution representing the dynamics of the MDP. Given an action $a \in \mathcal{A}$, the states have the Markov property, meaning that the past state is sufficient information for the next state.

3.2.2 Returns and Episodes

The interaction between the agent and the environment may end when a task is completed (the agent reaches a terminal state) or could last for infinite time steps. In this case, we talk about continuing tasks.

Definition 3.14. We call an **episode** a sequence of interactions described above between the agent and the environment. An episode has an initial time step (usually 0) and a final time step T, which may be infinite.

When trying to maximize the rewards we have to keep in mind that immediate actions may influence future rewards, even after many time steps. To better address this problem we introduce the following quantity:

Definition 3.15. The **discounted return** at time t is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.19}$$

where γ is the **discount factor**, the importance of future rewards at current time steps.

The discount factor is fundamental when dealing with infinite episodes, as the quantity in (3.19) is not well defined for $\gamma = 1$ and $T = \infty$. For example, if the reward is 1 at all time steps the series is divergent, while if $\gamma < 1$ we are sure that the overall return is bounded: $G_t \leq \max(R) \sum_{k=0}^{\infty} \gamma^k = \max(R) \frac{1}{1-\gamma}$. (recall that here $\mathcal{R}$ is finite). The discount factor also represents the importance we are giving to future rewards compared to more immediate ones. Using $\gamma = 1$ means that future rewards are as important as immediate ones while using $\gamma = 0$ means we are more myopic.

As a final remark, note that the final return of an episode depends on the transition and reward distributions and, most importantly for our analysis, on the policy of the agent.

3.2.3 Policies and Value functions

All Reinforcement Learning algorithms involve estimating and maximizing functions of states that indicate how good is for the agent to be in a state or to reach a state. Since future states and rewards depend on the actions the agent takes, these expected returns and rewards depend on the agent's policy.

Definition 3.16. A **policy** π is a mapping from the states space $\mathcal{S}$ to the space of probability distributions over $\mathcal{A}$. We indicate $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$.

Policies can be *deterministic*, meaning that for each state only one action is taken, or *stochastic* if, for each state, actions have an associated probability distribution.

Policies are the main strategy of an agent and are the main objective of learning. Agents explore the environment and by processing their interactions with it find the best policy to use. At the beginning of the exploration, policies are usually more randomized to gain knowledge about the environment but later become much more deterministic, once we exploit which actions are best to take in certain situations.

Now we need to define the targets to maximize during the learning process: these are policy-dependent quantities and involve the returns discussed above.

Definition 3.17. The **state-value function** $v_\pi(s)$ of a state s under a policy π is the expected return when starting from state s and following the policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{3.20}$$

where $\mathbb{E}_\pi$ means that we are considering subsequent actions taken with probabilities given by policy π.

Definition 3.18. The **action-value function** $q_\pi(s, a)$ of a state-action couple (s, a) under a policy π is the expected return when starting from state s, taking immediate action a and following the policy π afterward.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \tag{3.21}$$

where $\mathbb{E}_\pi$ means that we are considering subsequent actions taken with probabilities given by policy π.

Example 3.19. A possible way for evaluating the state value and action-value function is given by **Monte Carlo methods**: the agent is free to explore the environment using policy π for k episodes. Then, the state value function of state s is estimated by the average of the returns from every time step that s is visited. Let T_i be the total time steps of episode i and $G_t^{(i)}$ the return of episode i for time step t, then:

$$\begin{aligned} N(s) &= \sum_{i=1}^{k} \sum_{t=0}^{T_i} \mathbb{1}_{\{S_t^{(i)}=s\}} \\ \hat{v}_\pi(s) &= \frac{\sum_{i=1}^{k} \sum_{t=0}^{T_i} \mathbb{1}_{\{S_t^{(i)}=s\}} G_t^{(i)}}{N(s)} \end{aligned} \tag{3.22}$$

Similarly, for action value functions:

$$\begin{aligned} N(s,a) &= \sum_{i=1}^{k} \sum_{t=0}^{T_i} \mathbb{1}_{\{S_t^{(i)}=s, A_t^{(i)}=a\}} \\ \hat{q}_\pi(s,a) &= \frac{\sum_{i=1}^{k} \sum_{t=0}^{T_i} \mathbb{1}_{\{S_t^{(i)}=s, A_t^{(i)}=a\}} G_t^{(i)}}{N(s)} \end{aligned} \tag{3.23}$$

We now present the Bellman equations which give us relations between the action value and state value functions, as well as recursive expressions for both of them.

Proposition 3.1 (Bellman equations)**.** *The following recursive relations apply:*

$$\begin{aligned} v_\pi(s) &= \sum_{a\in\mathcal{A}(s)} \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \\ v_\pi(s) &= \sum_{a\in\mathcal{A}(s)} \pi(a|s) q_\pi(s,a) \\ q_\pi(s,a) &= \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'\in\mathcal{A}(s')} \pi(a'|s') q_\pi(s',a') \right] \\ q_\pi(s,a) &= \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s') \right] \end{aligned} \tag{3.24}$$

Intuitively, the state-value function of state s is given by the weighted average of action-value functions of (s,a) for the policy used. This quantity can be decomposed into the sum of immediate reward r and the discounted state value function of the successive state s', averaged over the MDP dynamics which determine the transition to the next step. The same reasoning applies similarly to q_π.

Proof. We will show the first two, the others are similar.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \sum_{a\in\mathcal{A}(s)} \pi(a|s) \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \sum_{a\in\mathcal{A}(s)} \pi(a|s) q_\pi(s,a) \end{aligned} \tag{3.25}$$

which gives us the second relation. Expanding one step further

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{r,s'} p(r, s'|s, a)[r + \gamma \mathbb{E}_{\pi G_{t+1}|S_{t+1}=s'}] \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{r,s'} p(r, s'|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{3.26}$$

□

This is important since it is possible to apply recursive methods to find the solution and verify the optimality of the policy. Solving an MDP means finding the best policy that achieves the best state-function value for states.

We can define a partial ordering on policies:

Definition 3.20. Given policies π, π' we say that $\pi \geq \pi'$ if and only if, for all states $s \in \mathcal{S}$, we have $v_\pi(s) \geq v_{\pi'}(s)$.

Definition 3.21. We call a policy π_* **optimal** if $v_{\pi_*} \geq v_\pi$ for all policies π.

There can be more than one optimal policy but from the definition it follows that all optimal policies have the same state-value and action-value functions:

Definition 3.22. The **optimal state-value function** is given by

$$v_*(s) = \max_\pi v_\pi(s) \tag{3.27}$$

The **optimal action-value function** is given by

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{3.28}$$

If we write the definition of the latter we can see that

$$\begin{aligned} q_*(s, a) &= \max_\pi \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \max_\pi \left(\mathbb{E}_\pi[R_{t+1} | S_t = s, A_t = a] + \gamma \mathbb{E}_\pi[G_{t+1} | S_t = s, A_t = a] \right) \\ &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a] + \gamma \max_\pi \mathbb{E}_\pi[G_{t+1} | S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a], \end{aligned} \tag{3.29}$$

where the expected value is taken with respect to MDP's dynamics. What we have just shown is one of the Bellman optimality equations for Markov Decision Processes. Results on the existence of an optimal policy, as well as a complete reformulation of these concepts in terms of normed spaces can be found in [Sze10].

Once the values for $q_\pi(s, a)$ and $v_\pi(s)$ are known it is straightforward to obtain an optimal policy. We simply need to act greedily: given state s, let $a^* \in \arg\max_a q_*(s, a)$ and set $\pi(a^*|s) = 1$ and $\pi(a|s) = 0$ otherwise. This policy is optimal as, at each step, we are taking the best possible action from the optimal action-value function, which considers future interactions as well.

Reinforcement Learning tasks involve two main problems: the evaluation of a policy π, which is finding the values for $v_\pi(s)$ and $q_\pi(s, a)$ for everything state and action and finding the best policy possible. These problems are sometimes addressed as evaluation and control respectively.

3.3 Model-free and Off-policy RL

In the last section, we analyzed Markov Decision Processes for Reinforcement Learning. In case the underlying MDP of the agent-environment interaction is known, we talk about model-based RL. In this case, the task can be completely solved with various techniques such as dynamic programming, since all the parameters and information on the process are known. Unfortunately, we don't always have access to all information needed to model the problem this way.

Constructing an MDP model and estimating its components using the agent's experience is possible, although the performance of the agent is limited by how accurate is the estimated model in representing the true underlying MDP. Model-based methods suffer also from the curse of dimensionality: a large number of states and actions implies lots of possibilities to be checked and a high computational load.

To solve these problems, model-free and approximation methods avoid direct modeling of the agent-environment interaction and instead try to learn an approximate mapping from the states to the state-value and action-value functions, which will be our main focus.

Definition 3.23. A **state-value function approximation** is a function of states which depends on some parameters $\boldsymbol{w}$ that approximates the true state-value function $\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s)$.

State-value function approximations can be of many kinds e.g. linear functions $\hat{v}(s; \boldsymbol{w}) = \boldsymbol{w}^T s$ or more complex mappings such as deep learning models. This approach is also useful to generalize between similar states, now represented by vectors and not by isolated components of the MDP.

The main error function to optimize is the *mean square value error* $\overline{VE}(\boldsymbol{w}) = \sum_{s \in \mathcal{S}} \mu(s) \left[v_\pi(s) - \hat{v}(s; \boldsymbol{w})\right]^2$, where $\mu(s)$ is a measure over states, sometimes the fraction of time spent in the state. Recall we want to approximate v_π, which is policy dependent. Many algorithms solve this optimization problem with gradient descent techniques: we gather episodes of experience by exploring the environment, collect transitions in batches and update the parameters

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left[v_\pi(S_t) - \hat{v}(S_t, \boldsymbol{w}_t)\right] \nabla \hat{v}(S_t, \boldsymbol{w}_t) \tag{3.30}$$

Notice that $v_\pi(S_t)$ is our ideal target but it is not available and needs to be replaced with a feasible objective function. Here are the two main examples:

- **Monte Carlo Estimation:** the agent explores the environment following the policy π, we collect the transitions and rewards, we calculate the returns G_t for each time step and we set as objective G_t. We are approximating the value function by full episodic experiences. For this to work all episodes must terminate in a finite number of time steps.

- **Temporal Difference Estimation TD(n):** the agent explores the environment following the policy π, we collect the transitions and rewards, the objective is set as $R_t + \sum_{k=1}^{n} \gamma^k R_{t+k} + \hat{v}(S_{t+n}; \boldsymbol{w})$. We are approximating the value function with a mixture of experience and our current belief on the value function approximation.

The method we use influences a lot the convergence of our approximation as we will see later.

Monte Carlo estimation has a low bias but high variance, since our actual target error $\overline{RE}(\boldsymbol{w}) = \mathbb{E}\left[(G_t - \hat{v}(S_t; \boldsymbol{w}))^2\right] = \mathbb{E}\left[(G_t - v_\pi(S_t))^2\right] + \overline{VE}(\boldsymbol{w})$ and the first term is a variance term due to stochastic exploration we cannot avoid. Temporal difference estimation reduces this variance by using our estimation, which is non-path-dependent, as part of the target. This has the cost of introducing bias since the target indeed depends on our current belief for $\hat{v}(S_t, \boldsymbol{w})$.

The same concepts apply **action-value function approximation**: in this case we want a function $\hat{q}(s, a; \boldsymbol{w}) \approx q_\pi(s, a)$. The only difference is that updates of the parameters are in this case action dependent. The objective function for the estimation of $TD(n)$ has to be chosen wisely: the next action has to be selected with respect to the policy we are estimating and not on the path the agent has taken during the exploration.

An example is the SARSA method [RN94], where the policy we follow is called ϵ-greedy:

$$\pi_\epsilon(a|s) = \begin{cases} 1 - \epsilon \text{ if } a = \arg\max_{a'} \hat{q}(s, a'; \boldsymbol{w}) \\ \frac{\epsilon}{m-1} \text{ otherwise} \end{cases} \tag{3.31}$$

where m is the number of possible actions from s. In this case, the target function for the parameter update is $R_t + \gamma \hat{q}(S_{t+1}, A'; \boldsymbol{w})$ where A' is sampled using the same ϵ-greedy policy described above. This technique of using the same policy for exploration and parameter update is called **on-policy RL**.

We refer to **off-policy RL** whenever we update the action-value and state-value functions relative to a policy π while using another policy $b(a|s)$ for exploration. This way we can evaluate greedy policies which, for discrete and finite environments, always follow the best action at each time step but cannot be used for exploration, which is usually performed with an ϵ-greedy policy.

Most off-policy Reinforcement Learning algorithms with temporal difference learning use a **replay buffer**, a set containing all previous experience of the agent interacting with the environment in the form of transitions $S_t, A_t, R_{t+1}, S_{t+1}$. Choosing samples of experience uniformly from the replay buffer improves learning performance [MKS+15] as opposed to using consecutive actions and experience, which introduces high correlations between the samples. Another advantage is that the algorithms have an increased number of samples that can be used to improve the action and state value function approximations instead of using every transition once. The replay buffer usually has a maximum size: if this value is reached, the earliest transitions are removed to make space for the newer ones.

We have seen the evaluation of a policy π both with a model-based and a model-free approach. If we alternate the evaluation of a policy and the subsequent calculation of the greedy policy we obtain the **policy evaluation** methods. With the right hypothesis, one can show that the greedy policy converges to the optimal one (e.g. [SJLS00]). Unfortunately, the combination of action-value approximation, off-policy RL and TD-learning is an example of a policy iteration method that not always converges to the optimal policy.

3.3.1 Policy Gradient Methods

Previously shown methods are all considered as **action-value methods** since they learn the value function and determine the best deterministic policy following a greedy approach. In this section, we introduce methods that parametrize the policy, called

policy gradient methods, that can be used coupled with an action-value function or as a standalone.

We indicate the policy parameters as $\theta \in \mathbb{R}^{d'}$, so that we can express the policy as $\pi(a|s,\theta) = \mathbb{P}[A_t = a|S_t = s, \theta_t = \theta]$. To learn the optimal θ we need an objective function that measures the performance of the policy. This function is usually indicated as $J(\theta)$. Policy optimization is done by approximate gradient methods, updating the parameters as $\theta_{t+1} = \theta_t + \alpha\hat{\nabla J(\theta_t)}$, where the last term is a stochastic approximation of the true gradient evaluated at current parameters.

Parametrized policies can be deterministic or stochastic, even though to ensure exploration we require them not to be deterministic at least at the beginning of training. They are also very flexible and can be applied in continuous action and state spaces, with the only true requirement being for the policy to be differentiable with a finite gradient.

An example of policy approximation with finite a finite action space is the softmax policy $\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$ where $h(s,a,\theta)$ is a function that represents action preferences: the greater $h(s,a,\theta)$ with respect to a, the greater is the probability to chose this action. These action preferences can be represented by the action-value function or any differentiable function such as linear functions or Neural Networks. In the later stages of the learning procedure close to convergence, we can switch to a more deterministic policy to maximize the rewards. For example, we can let $\epsilon \to 0$ in an ϵ-greedy policy.

A widely used performance function is the state-value function of the initial state $J(\theta) = v_{\pi_\theta}(S_0)$, where $v_{\pi_\theta}(s)$ is the true state-value function of the policy. The gradient of $J(\theta)$ depends on both the policy itself and the dynamics of the environment. However, theorem 3.2 provides an analytic expression for these quantities.

Theorem 3.2 (Policy Gradient Theorem [Wil92])**.** *Assume that the action and state spaces and trajectories are finite, let $\mu(s)$ be the marginal distribution of being in a state s following the policy π_θ. Assume also that $\gamma = 1$ Then, the gradient of the performance function $J(\theta) = v_{\pi_\theta}(s)$ is proportional to:*

$$\nabla J(\theta) \propto \sum_{s\in\mathcal{S}} \mu(s) \sum_{a\in\mathcal{A}} q_{\pi_\theta}(s,a)\nabla\pi(a|s,\theta) \tag{3.32}$$

Proof. We start by calculating the gradient of the state value function for a particular

state s

$$\begin{aligned}
\nabla v_{\pi_\theta}(s) &= \nabla \left[\sum_a \pi_\theta(a|s) q_{\pi_\theta}(s,a) \right] \\
&= \sum_a [\nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla q_{\pi_\theta}(s,a)] \\
&= \sum_a \left[\nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \nabla \sum_{s',r} p(s',r|s,a)(r + \gamma v_{\pi_\theta}(s')) \right] \\
&= \sum_a \left[\nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \gamma \sum_{s'} p(s'|s,a) \nabla v_{\pi_\theta}(s') \right] \\
&= \sum_a \left[\nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) + \pi_\theta(a|s) \gamma \sum_{s'} p(s'|s,a) \right. \\
&+ \left. \sum_{a'} \left[\nabla \pi_\theta(a'|s') q_{\pi_\theta}(s',a') + \pi_\theta(a'|s') \sum_{s''} p(s''|s',a') \nabla v_{\pi_\theta}(s'') \right] \right]
\end{aligned} \tag{3.33}$$

Define $\Pr[s \to x, k, \pi_\theta]$ as the probability of being in state x after k steps starting from s and following π_θ. The above expression is then

$$\nabla v_{\pi_\theta}(s) = \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr[s \to x, k, \pi_\theta] \sum_a \nabla \pi_\theta(a|x) q_{\pi_\theta}(x,a) \tag{3.34}$$

Hence,

$$\begin{aligned}
\nabla J(\theta) &= \nabla v_{\pi_\theta}(S_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr[S_0 \to s, k, \pi_\theta] \right) \sum_a \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_s \nu(s) \sum_a \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_{s'} \nu(s') \sum_s \frac{\nu(s)}{\sum_{s'} \nu(s')} \sum_a \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&= \sum_{s'} \nu(s') \sum_s \mu(s) \sum_a \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a)
\end{aligned} \tag{3.35}$$

□

An example of a policy gradient method is REINFORCE [Wil92]. To obtain its

update, we start from the gradient formula we have just obtained

$$\begin{aligned}
\nabla J(\theta) &= \sum_s \mu(s) \sum_a q_{\pi_\theta}(s,a) \nabla \pi_\theta(a|s) \\
&= \mathbb{E}_{\pi_\theta} \left[\sum_a q_{\pi_\theta}(S_t, a) \nabla \pi_\theta(a|S_t) \right] \\
&= \mathbb{E}_{\pi_\theta} \left[\sum_a \pi_\theta(a|S_t) q_{\pi_\theta}(S_t, a) \frac{\nabla \pi_\theta(a|S_t)}{\pi_\theta(a|S_t)} \right] \\
&= \mathbb{E}_{\pi_\theta} \left[q_{\pi_\theta}(S_t, A_t) \frac{\nabla \pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \right] \quad \text{with } A_t \sim \pi_\theta \\
&= \mathbb{E}_{\pi_\theta} \left[G_t \frac{\nabla \pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \right] \quad \text{for the definition of } q_{\pi_\theta}
\end{aligned} \tag{3.36}$$

Given this, we define

$$\theta_{t+1} = \theta_t + \alpha \gamma^t G_t \frac{\nabla \pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \tag{3.37}$$

Intuitively, the gradient follows the direction which favors the repetition of action A_t being in state S_t normalized to the current probability of selecting that action, to prevent the most selected ones from having an advantage. The update in that direction is also proportional to the return, so the most promising actions are preferred.

Since REINFORCE is a Monte Carlo method it tends to have high variance. To avoid this problem we can introduce a baseline function $b(s)$ to reduce the variance at the cost of some bias.

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_{\pi_\theta}(s,a) - b(s)) \nabla \pi_\theta(a|s) \tag{3.38}$$

with update equation

$$\theta_{t+1} = \theta_t + \alpha \gamma^t (G_t - b(S_t)) \frac{\nabla \pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \tag{3.39}$$

The above formula for the gradient is correct although we have introduced the baseline term since if we isolate it we obtain

$$\sum_a b(s) \nabla \pi_\theta(a|s) = b(s) \nabla \sum_a \pi_\theta(a|s) = b(s) \nabla 1 = 0 \tag{3.40}$$

One natural choice for the baseline is to use the state-value function $b(s) = v_{\pi_\theta}(s)$ or, more practically, an estimate of it. The update of the parameter is proportional to the gain for an action with respect to the current belief on the state-value function. Furthermore, the baseline helps reduce the norm of the gradient, acting as a regularization while introducing a bias.

3.3.2 Actor-Critic methods

In the REINFORCE algorithm, the baseline function is used to assess just the first state of the transition. The update involves the complete return, making it an offline

method and not computationally efficient in general. As before, we introduce the Temporal Difference update in this framework obtaining the following update rule

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha(G_{t:t+1} - v_{\pi_\theta}(S_t))\frac{\nabla\pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \\ &= \theta_t + \alpha(R_{t+1} + \gamma v_{\pi_\theta}(S_{t+1}) - v_{\pi_\theta}(S_t))\frac{\nabla\pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)}\end{aligned} \tag{3.41}$$

These models are called **Actor-Critic methods** since the update for the parameters is assessed by the state-value function which plays the role of a *critic*.

Actor-Critic methods with temporal difference updates can be efficiently performed in an online manner with a state-value function approximation $\hat{v}(s, w)$ as baseline function. The full procedure is described in algorithm 3

Algorithm 3 One step Actor-Critic algorithm

```
Initialize parameters θ, w
for each each episode do
    Initialize I = 1, initial state S_0
    for each environmental step do
        A_t ~ π_θ(A_t|S_t)
        S_{t+1} ~ p(S_{t+1}|S_t, A_t)
        δ_t ← R_{t+1} + γv̂(S_{t+1}, w) − v̂(S_t, w)
        w ← w + α_w δ ∇v̂(S_t, w)
        θ ← θ + α_θ I δ ∇ ∇π_θ(A_t|S_t) / π_θ(A_t|S_t)
        I ← γI
    end for
end for
```

3.4 Soft Actor-Critic

In this section, we present the main method which will serve as a basic model for the main algorithm used to solve the CFT bootstrapping problem.

In addition to all the previous notions about Markov decision processes, we denote with $\rho_\pi(S_t)$ and $\rho_\pi(S_t, A_t)$ respectively the state and state-action marginals of the distribution over trajectory, as induced by the policy $\pi(A_t|S_t)$.

Definition 3.24. Let X be a discrete random variable that takes values in $\mathcal{X}$. Let $p(x)$ the distribution of X, meaning that $p(x) = \mathbb{P}[X = x]$. The entropy of X is defined as

$$\mathcal{H}(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) \tag{3.42}$$

The same definition applies to conditional distributions and any distribution $p(x)$ in general with the notation $\mathcal{H}(p)$.

Definition 3.25. Let X be a random variable on a measurable space $\mathcal{X}$ with values in $\mathbb{R}$. Let $\mathbb{P}_X$ be its probability distribution and assume it has a density $f(x)$. The **entropy** of X is defined as:

$$\mathcal{H}(X) = -\int_{\mathbb{R}^d} f(x)\log f(x)\mathrm{d}x \tag{3.43}$$

The definition also applies to any distribution density in general.

Entropy is the most important concept in information theory, introduced years ago by Shannon in [Sha48], widely used in many applications such as telecommunication, coding, ML and probability. Entropy represents the uncertainty of sampling a random variable or distribution: a low value means we are almost certain of the outcome of our process while a higher entropy value is a sign of maximum uncertainty. In fact, for discrete and finite spaces the maximum entropy distribution is the uniform one. We refer to [CT12] for a general discussion on entropy, its properties and applications.

We assume the reward is bounded in $[r_{\min}, r_{\max}]$ and it is a deterministic function of the current state and action taken $r(S_t, A_t)$, which will be the case in our formulation of the problem, sometimes abbreviated as $r_t = r(S_t, A_t)$.

While standard Reinforcement Learning techniques aim to maximize the expected return, in our case, we use a more general objective involving entropy as well:

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(S_t,A_t)\sim\rho_\pi} \left[\sum_{l=t}^{\infty} \gamma^{l-t} \mathbb{E}_{S_l\sim p_d, A_l\sim\pi} \left[r(S_l, A_l) + \alpha \mathcal{H}(\pi(\cdot|S_l)) \middle| S_t, A_t \right] \right] \tag{3.44}$$

We want to maximize the expected return and entropy of our policy, averaged over the state-action marginal of the trajectory distribution. This new objective function improves exploration [HTAL17a, SAC17] and, in some cases, the speed of learning [HZAL18]. The α parameter is a temperature parameter that determines the relative importance of the entropy against the reward. This controls the stochasticity of our policy and the objective is to incentivize exploration while avoiding poor rewards. We can assume $\alpha = 1$: this parameter can be integrated into the reward by scaling it with a factor α^{-1} as the solution to the objective above remains the same (simply group by a factor α).

3.4.1 Soft Policy Iteration

For now, we adopt the same MDP setting as before, assuming the sets of states and actions are finite to prove some results. Soon, we will generalize this approach to continuous spaces and policies. We also assume $\gamma < 1$.

To understand the soft actor-critic method we need to introduce soft value functions and energy-based models [HTAL17b], in which policies take the form $\pi(A_t|S_t) = \exp(-\mathcal{E}(S_t, A_t))$, where $\mathcal{E}(A, S)$ is an energy function. These models were introduced to build and develop a general framework for stochastic policies.

Call π^*_{maxEnt} the solution to the maximization problem of 3.44.

Definition 3.26. The **soft Q-function** is defined as

$$Q^\pi_{\text{soft}}(S_t, A_t) = r_t + \mathbb{E}_{S_{t+1},\ldots\sim\rho_\pi} \left[\sum_{l=1}^{\infty} \gamma^l (r_{t+l} + \mathcal{H}(\pi(\cdot|S_{t+l}))) \right] \tag{3.45}$$

The **soft value-function** is defined as

$$V^\pi_{\text{soft}}(S_t) = \mathbb{E}_{A_t\sim\pi} \left[Q^\pi_{\text{soft}}(S_t, A_t) - \log \pi(A_t|S_t) \right] \tag{3.46}$$

Define also $V^*_{\text{soft}}(\cdot)$ and $Q^*_{\text{soft}}(\cdot,\cdot)$ as the soft functions for the π^*_{maxEnt} policy.

The soft policy iteration step is given by the following operator (soft Bellman operator) on functions $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

$$\mathcal{T}^\pi Q(S_t, A_t) \triangleq r(S_t, A_t) + \gamma \mathbb{E}_{S_{t+1} \sim p}[V(S_{t+1})] \tag{3.47}$$

where V is defined as the soft value function for the function Q.

Theorem 3.3 (Soft Policy Evaluation [HTAL17b])**.** *Given any function $Q^0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, define the sequence $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then, Q^k converges to the soft Q-function of π Q^π_{soft} as $k \to \infty$, provided that $\gamma < 1$.*

Proof. First, we show that the soft Q-function for π is a fixed point for the soft Bellman operator for π.

$$\begin{aligned}
\mathcal{T}^\pi Q^\pi_{\text{soft}}(S_t, A_t) &= r_t + \gamma \mathbb{E}_{S_{t+1} \sim p, A_{t+1} \sim \pi} \left[r_{t+1} + \mathbb{E}_{\rho_\pi} \left[\sum_{l=2}^{\infty} \gamma^l (r_{t+l} + \mathcal{H}(\pi(\cdot|S_{t+1}))) \right] \right] \\
&= r_t + \gamma \mathbb{E}_{S_{t+1}, A_{t+1}, \ldots \sim \rho_\pi} \left[\sum_{l=2}^{\infty} \gamma^l (r_{t+l} + \mathcal{H}(\pi(\cdot|S_{t+1}))) \right] \\
&= Q^\pi_{\text{soft}}(S_t, A_t)
\end{aligned} \tag{3.48}$$

On the space of functions $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ consider the norm $\|Q\| = \max_{s,a} |Q(s,a)|$. It suffices to show that the Bellman operator is a contraction. Let Q_1, Q_2 any functions as above, $\epsilon = \|Q_1 - Q_2\|$. Then

$$\begin{aligned}
\|\mathcal{T}^\pi Q_1 - \mathcal{T}^\pi Q_2\| &= r_t + \gamma \mathbb{E}_{\rho_\pi} \left[Q_2(S_{t+1}, A_{t+1}) - \log \pi(A_{t+1}|S_{t+1}) \right] \\
&\quad - r_t - \gamma \mathbb{E}_{\rho_\pi} \left[Q_2(S_{t+1}, A_{t+1}) - \log \pi(A_{t+1}|S_{t+1}) \right] \\
&= \gamma \mathbb{E}_{S_{t+1} \sim p, A_{t+1} \sim \pi} \left[Q_1(S_{t+1}, A_{t+1}) - Q_2(S_{t+1}, A_{t+1}) \right] \\
&\leq \gamma \mathbb{E}_{S_{t+1} \sim p, A_{t+1} \sim \pi} \left[\|Q_1 - Q_2\| \right] \\
&= \gamma \|Q_1 - Q_2\| = \gamma \epsilon
\end{aligned} \tag{3.49}$$

which is smaller than 1 provided that $\gamma < 1$. □

Once the soft Q-function is obtained we need to improve the policy itself. To improve future tractability we restrict to a class of stochastic policies Π (for example, later we will use Gaussians). This requires us to project at each step the improved policy on this set. This is done using the Kullback-Leibler divergence:

Definition 3.27. Let $p(x), q(x)$ be two distributions over a discrete set $\mathcal{X}$. The **Kullback-Leibler divergence** is defined as

$$D_{\text{KL}}(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{3.50}$$

Definition 3.28. Let $f(x), g(x)$ be any probability density functions on a measurable space $\mathcal{X}$. The **Kullback-Leibler divergence** is defined as

$$D_{\text{KL}}(f\|g) = \int_{\mathcal{X}} f(x) \log \frac{f(x)}{g(x)} \mathrm{d}x \tag{3.51}$$

The updated policy is then given by

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot|S_t) \middle\| \frac{\exp(Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, \cdot))}{Z^{\pi_{\text{old}}}(S_t)} \right) \tag{3.52}$$

where $Z^{\pi_{\text{old}}}$ is a partition function to normalize. Although this is not tractable in general this term does not contribute to the gradient for the new policy and will be ignored afterward.

Let's show that the policy is indeed improving.

Theorem 3.4 (Soft policy Improvement [HTAL17b]). *Let $\pi_{old} \in \Pi$ and let $\pi_{new} \in \Pi$ given by equation (3.52). Then $Q^{\pi_{new}}_{soft}(S_t, A_t) \geq Q^{\pi_{old}}_{soft}(S_t, A_t)$ for all $(S_t, A_t) \in \mathcal{S} \times \mathcal{A}$ provided that $|\mathcal{A}| < \infty$.*

Proof. Consider

$$\begin{aligned} \pi_{\text{new}}(\cdot|S_t) &= \arg\min_{\pi' \in \Pi} D_{\text{KL}}(\pi'(\cdot|S_t) | \exp(Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, \cdot) - \log Z^{\pi_{\text{old}}}(S_t))) \\ &= \arg\min_{\pi' \in \Pi} I_{\pi_{\text{old}}}(\pi'(\cdot|S_t)) \end{aligned} \tag{3.53}$$

We have surely that $I_{\pi_{\text{old}}}(\pi_{\text{new}}) \leq I_{\pi_{\text{old}}}(\pi_{\text{old}})$. From the definition of the KL-divergence, we have that

$$\mathbb{E}_{\pi_{\text{new}}}[\log \pi_{\text{new}}(A_t|S_t) - Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, A_t) + \log Z^{\pi_{\text{old}}}(S_t)] \leq \tag{3.54}$$

$$\leq \mathbb{E}_{\pi_{\text{old}}}[\log \pi_{\text{old}}(A_t|S_t) - Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, A_t) + \log Z^{\pi_{\text{old}}}(S_t)] \tag{3.55}$$

Since the partition function doesn't depend on the policy, the term vanishes on both sides giving us

$$\mathbb{E}_{\pi_{\text{new}}}[Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, A_t) - \log \pi_{\text{new}}(A_t|S_t)] \geq V^{\pi_{\text{old}}}_{\text{soft}}(S_t) \tag{3.56}$$

From Bellman's equation, we have that

$$\begin{aligned} Q^{\pi_{\text{old}}}_{\text{soft}}(S_t, A_t) &= r_t + \gamma \mathbb{E}_{S_{t+1} \sim p}[V^{\pi_{\text{old}}}_{\text{soft}}(S_t)] \\ &\leq r_t + \gamma \mathbb{E}_{S_{t+1} \sim p}[\mathbb{E}_{\pi_{\text{new}}}[Q^{\pi_{\text{old}}}_{\text{soft}}(S_{t+1}, A_{t+1}) - \log \pi_{\text{new}}(A_{t+1}|S_{t+1})]] \\ &= r_t + \gamma \mathbb{E}_p[\mathbb{E}_{\pi_{\text{new}}}[-\log \pi_{\text{new}}(A_{t+1}|S_{t+1}) + r_{t+1} + \mathbb{E}_{S_{t+2} \sim p}[V^{\pi_{\text{old}}}_{\text{soft}}(S_{t+2})]]] \\ &\vdots \\ &\leq r_t + \mathbb{E}_{\rho_{\pi_{\text{new}}}} \sum_{l=1}^{\infty} \gamma^l (r_{t+l} + \mathcal{H}(\pi_{\text{new}}(\cdot|S_{t+l}))) = Q^{\pi_{\text{new}}}_{\text{soft}}(S_t, A_t) \end{aligned} \tag{3.57}$$

□

The main algorithm for soft policy iteration repeatedly alternates between soft policy evaluation and improvement, until it convergence to the optimal policy in the set Π.

Theorem 3.5 (Soft policy convergence [HTAL17b]). *Starting from any $\pi' \in \Pi$, the algorithm that alternates soft policy evaluation and improvement converges to a policy π^* such that $Q^*_{soft}(S_t, A_t) \geq Q^{\pi}_{soft}(S_t, A_t)$ for any $\pi \in \Pi$ and any $(S_t, A_t) \in \mathcal{S} \times \mathcal{A}$, provided that $|\mathcal{A}| < \infty$.*

Proof. Let π_i the policy at iteration i of the algorithm, by theorem 3.4 the sequence $Q_{\text{soft}^{\pi_i}}$ is monotonically increasing. Since we are assuming bounded rewards and the entropy is bounded ($|\mathcal{A}| < \infty$), the sequence converges to π^*. We have to show that π^* is optimal. Since at each step we are minimizing I_π as in the theorem above, we have that at convergence $I_{\pi^*} \leq I_\pi$ for all $\pi \in \Pi$. Using the same iterative argument as the last theorem, we have $Q^*_{\text{soft}}(S_t, A_t) \geq Q^\pi_{\text{soft}}(S_t, A_t)$ for all $(S_t, A_t) \in \mathcal{S} \times \mathcal{A}$. Since this is valid for any $\pi \in \Pi$ the policy π^* is optimal in Π. □

3.4.2 Soft Actor-Critic

Soft Actor-Critic [HZAL18] considers continuous action and state spaces. We cannot apply the same techniques described above, since the majority of those methods work in a tabular setting, where variables are discrete.

To operate in this setting we need to use state and action value functions approximations and gradient methods. The fundamental idea is to adopt Neural Networks as function approximations since they can learn complex functions while also being very flexible and differentiable. In any case, the following discussion applies to any differentiable function approximation method.

Consider the soft state value function, now written in a simple way as $V_\psi(S_t)$, and its parameters ψ (for example the weights and biases of a deep neural network), the soft Q-value function $Q_\theta(S_t, A_t)$ with parameters θ and a policy $\pi_\phi(A_t|S_t)$ with parameters ϕ. We also require the policy $\pi_\phi \in \Pi$ to be tractable, taking for example a Gaussian with mean and variance given by the outputs of a neural network.

In theory, there is no need to have both a soft value function and a soft Q-function as they are related through equation 3.46. In practice, introducing this additional bias can improve the stability of the training procedure [HZAL18].

The objective to be minimized by the soft value approximation is

$$J_V(\psi) = \mathbb{E}_{S_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(S_t) - \mathbb{E}_{A_t \sim \pi_\phi} [Q_\theta(S_t, A_t) - \log \pi_\phi(A_t|S_t)] \right)^2 \right] \tag{3.58}$$

where $\mathcal{D}$ is a distribution over previously sampled states. Soft Actor-Critic uses a replay buffer to store samples of experience.

To calculate the gradient of equation (3.58) we use the following estimator:

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(S_t)(V_\psi(S_t) - Q_\theta(S_t, A_t) + \log \pi_\phi(A_t|S_t)) \tag{3.59}$$

but instead of using the distribution over experience or the replay buffer, A_t is now sampled from the current policy to follow the actual behavior of the agent.

For the soft Q-function, the main objective is to match the soft Bellman residual update

$$\begin{aligned} J_Q(\theta) &= \mathbb{E}_{(S_t, A_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(S_t, A_t) - \hat{Q}(S_t, A_t) \right)^2 \right] \\ \hat{Q}(S_t, A_t) &= r(S_t, A_t) + \gamma \mathbb{E}_{S_{t+1} \sim p} [V_{\bar{\psi}}(S_t)] \end{aligned} \tag{3.60}$$

where in this case the stochastic gradient is given by

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(S_t, A_t)(Q_\theta(S_t, A_t) - r(S_t, A_t) - \gamma V_{\bar{\psi}}(S_{t+1})) \tag{3.61}$$

We are not using the current value of the parameters ψ in the calculation of the soft Q-function gradient, instead $\bar{\psi}$ is a weighted moving average of the ψ parameter. This improves the stability of training [MKS+15], as a continuous change in the ψ parameters leads to the gradient following an objective that continuously varies. The idea comes from the Deep Q-Network algorithm [MKS+15], where the objective to match for the soft Q-function is obtained by older θ parameters.

For the policy parameters, we recall we want to minimize the KL divergence as

$$J_\pi(\phi) = \mathbb{E}_{S_t \sim \mathcal{D}} \left[D_{\mathrm{KL}} \left(\pi_\phi(\cdot|S_t) \middle\| \frac{\exp(Q_\theta(S_t, \cdot))}{Z_\theta(S_t)} \right) \right] \tag{3.62}$$

Minimizing this quantity with a general stochastic policy for gradient policy methods can be complex and is usually done via likelihood gradient estimators [Wil92]. Since our policy is given by the soft Q-function and the latter is represented by a neural network we can directly differentiate and backpropagate easily.

We still need to apply the so-called **Reparametrization trick**: the action is given by $A_t = f_\phi(\epsilon_t, S_t)$, where ϵ_t is a noise vector, usually sampled from a spherical Gaussian. A common choice for f is $f_\phi(\epsilon_t, S_t) = \mu_\phi(S_t) + \epsilon_t \sigma_\phi(S_t)$. This has two advantages: we can model any Gaussian by sampling from a single distribution and the components that need to be differentiated are purely deterministic. $\mu_\phi(S_t), \sigma_\phi(S_t)$ are represented by a neural network.

Indeed, now the objective (3.62) can be written as

$$J_\pi(\phi) = \mathbb{E}_{S_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}}[\log \pi_\phi(f_\phi(\epsilon, S_t)|S_t) - Q_\theta(S_t, f_\phi(\epsilon_t, S_t))] \tag{3.63}$$

where we have omitted the partition function since it depends only on θ. The gradient is approximated as

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(A_t|S_t) + (\nabla_{A_t} \log \pi_\phi(A_t|S_t) - \nabla_{A_t} Q(S_t, A_t)) \nabla_\phi f_\phi(\epsilon_t, S_t) \tag{3.64}$$

where A_t is evaluated at $f_\phi(\epsilon_t, S_t)$. This is a generalization of the Deep Deterministic Policy Gradient algorithm (DDPG [LHP+16]) that applies also for stochastic policies.

Double Q-Networks

As shown in [FvHM18], Q-learning methods suffer from the so-called *positive bias*. Our soft Q-function is just an approximation of the true function and, even a 0-mean error in this estimation can lead to consistent overestimation in bias. For the following discussion we will use the policy gradient update from the DDPG method, the policy is then deterministic $A_t = \pi_\phi(S_t)$.

Give current policy parameters π, let ϕ_{approx} be the updated parameters induced by the maximization of the approximate critic $Q_\theta(s, a)$.

$$\phi_{\mathrm{approx}} = \phi + \frac{\alpha}{Z_1} \mathbb{E}_{s \sim \rho_\pi}[\nabla_\phi \pi_\phi(s) \nabla_a Q_\theta(s, a)|a = \phi(s)] \tag{3.65}$$

Define also ϕ_{true} as the new parameters from the purely theoretical actor update based on the true Q-function $Q^\pi(s, a)$.

$$\phi_{\mathrm{true}} = \phi + \frac{\alpha}{Z_2} \mathbb{E}_{s \sim \rho_\pi}[\nabla_\phi \pi_\phi(s) \nabla_a Q^\pi(s, a)|a = \phi(s)] \tag{3.66}$$

where Z_1 and Z_2 are normalizations of the expected values. Define $\pi_{\text{true}}, \pi_{\text{approx}}$ as the policies given by those parameters. Since these follow the gradient of their respective Q-functions, we have that for ϵ_1 sufficiently small, $\forall \alpha \leq \epsilon_1$

$$\mathbb{E}_{s\sim\rho_\pi}[Q_\theta(s, \pi_{\text{approx}}(s))] \geq \mathbb{E}_{s\sim\rho_\pi}[Q_\theta(s, \pi_{\text{true}}(s))] \tag{3.67}$$

and, on the other hand, we can find ϵ_2 sufficiently small such that $\forall \alpha \leq \epsilon_2$

$$\mathbb{E}_{s\sim\rho_\pi}[Q^\pi(s, \pi_{\text{true}}(s))] \geq \mathbb{E}_{s\sim\rho_\pi}[Q^\pi(s, \pi_{\text{approx}}(s))] \tag{3.68}$$

Furthermore, if $\mathbb{E}_{s\sim\rho_\pi}[Q_\theta(s, \pi_{\text{true}}(s))] \geq \mathbb{E}_{s\sim\rho_\pi}[Q^\pi(s, \pi_{\text{true}}(s))]$, the two equations imply that

$$\mathbb{E}_{s\sim\rho_\pi}[Q_\theta(s, \pi_{\text{approx}}(s))] \geq \mathbb{E}_{s\sim\rho_\pi}[Q^\pi(s, \pi_{\text{approx}}(s))] \tag{3.69}$$

for any value of $\alpha \leq \min(\epsilon_1, \epsilon_2)$. This means that our critic is in reality an overestimation of the true Q-function. This error may propagate even more through some iteration and lead to poor policy updates.

The proposed solution to this issue is to use two soft Q-functions instead of one $Q_{\theta_1}, Q_{\theta_2}$ [FvHM18], trained independently to optimize the two objectives $J_Q(\theta_1), J_Q(\theta_2)$. Then, the Bellman update given by $r_t + \gamma Q_\theta(s, \pi_\phi(s))$ is substituted by its final form

$$r_t + \gamma \min_{j=1,2} Q_{\theta_j}(s, \pi_\phi(s)). \tag{3.70}$$

These additional features improve again the stability and the learning speed of the algorithm [FvHM18].

SAC algorithm

The complete pseudocode for SAC can be found in algorithm 4.

Algorithm 4 Soft Actor-Critic algorithm

1: Initialize parameters $\psi, \bar{\psi}, \theta_j, \phi$
2: **for** each each iteration **do**
3: **for** each environmental step **do**
4: $A_t \sim \pi_\phi(A_t|S_t)$
5: $S_{t+1} \sim p(S_{t+1}|S_t, A_t)$
6: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S_t, A_t, r(S_t, A_t), S_{t+1})\}$
7: **end for**
8: **for** each gradient step **do**
9: $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
10: $\theta_j \leftarrow \theta_j - \lambda_Q \hat{\nabla}_{\theta_j} J_Q(\theta_i)$ for $j = 1, 2$
11: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
12: $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
13: **end for**
14: **end for**

The algorithm alternates exploration and exploitation following the processes described previously. The agent explores the environment for a step (lines 4 and 5) and the transition gets stored in the replay buffer (6). Then, we improve the performance of all the neural networks involved as approximators: the value network (line 9), both

Q-networks (line 10) and the actor network (line 11). Optimization of such networks is performed by some gradient steps on samples extracted from the replay buffer.

Each set of parameters for the neural networks is updated using possibly different learning rates: $\lambda_V, \lambda_Q, \lambda_\pi$. The ψ parameter, related to the value network, is also exponentially smoothed with an update factor τ. The actual parameters for the neural network update are calculated as $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$ (line 12) to improve the stability of the estimated value function.

Chapter 4

Bootstrap Stochastic Optimization with SAC

In this Chapter, we will focus on developing an RL-based approach for conformal bootstrap, as well as presenting the two models we will apply this framework to: the $2D$ Ising model and the $1D$ defect CFT defined on the $\frac{1}{2}$-BPS Wilson line in the $N = 4$ super Yang-Mills theory.

The specific RL algorithm used is a modification of Soft Actor-Critic, first presented in [KPN22a], [KPN22b] and then updated in [KNPR23] to improve efficiency and precision, as well as automate some subtasks previously done manually.

The $2D$ Ising model will be used to validate this approach since an analytical solution to the crossing equation is known, while the one-dimensional model represents the innovative result of this thesis. In fact, no applications of RL to this model exist before this. To do so, we need to adapt the algorithm to the one-dimensional CFT by defining a new RL environment. This is an additional difficulty since it is known that applying the same RL algorithm to different problems is not straightforward.

Furthermore, this new model has two additional constraints on the 4-point function [CGJP22a] which can be used to increase the precision in the estimates of the unknowns in the CFT. We search for ways to implement these constraints into the framework with some reward engineering. Finally, we show previous results that were able to find the values for the scaling dimension of the first 10 operators for the one-dimensional CFT with arbitrary precision, which can be used as input to reduce the complexity of the problem.

The $2D$ Ising model is again a useful tool to test the possibility of including known values as input.

4.1 Reinforcement learning approach for Conformal Bootstrap

In this section, we present the main algorithm for the stochastic optimization of the conformal bootstrap using SAC. The original implementation comes from [KNPR23] while we adapted this algorithm to the two models studied as well as automating the experiment with an increasing number of scaling dimensions fixed.

Our main objectives are the four-point functions and, in particular, the conformal bootstrap equation in (2.140). Using the $z, \bar{z}$ coordinates in the complex plane we

rewrite this equation as

$$\sum_i C_i^2 F_{\Delta_i, s_i}(z, \bar{z}) = 0 \tag{4.1}$$

where C_i^2 is the squared OPE coefficient and the sum runs over the spectrum of local operators of the theory, labeled by their scaling dimensions Δ_i and spins $s_i \in \mathbb{N}$. Note that since the group of rotation is trivial on a line, there is no spin for the one dimensional case. Our set of unknowns is given by the CFT data $\{C_i^2, \Delta_i, s_i\}$. This set is infinite and it is therefore infeasible to search for all the values if no analytical solution is available.

A remarkable feature of CFT is that the operator product expansion converges exponentially in Δ [PRER12, Pol98]. Hence, we do not search for an exact solution to the problem, involving an infinite number of variables, but we instead truncate the expansion at a cutoff $\Delta_{\max}$, searching for the values of C_i^2, Δ_i such that $\Delta_i \leq \Delta_{\max}$. The exponential convergence ensures that the above equation (4.1) will not be solved exactly but up to an exponentially small factor:

$$\sum_{i:\Delta_i \leq \Delta_{\max}} C_i^2 F_{\Delta_i, s_i}(z, \bar{z}) = O(e^{-\Delta_{\max}}) \tag{4.2}$$

To apply reinforcement learning to the bootstrap equation we need to define all the components of the general framework:

The environment

Every reinforcement learning algorithm requires an environment that guides the learning of the agent to predict the CFT data and is therefore the most important part of the framework. The environment in which the actor moves is constituted by the space of the parameters

$$(s_1, s_2, \ldots, s_n, \Delta_1, \Delta_2, \ldots, \Delta_n, C_1^2, C_2^2, \ldots, C_n^2) \in [0, s_{\max}] \times [0, \Delta_{\max}]^n \times [0, \infty)^n \tag{4.3}$$

With this being the most general setting we can think of for the approach, it is useful to make some particular remarks on the implementation we actually need:

- The models we are working with, being the $2D$ Ising model and the $1D$ defect CFT considered, let us drastically simplify the environment for what concerns the spins s_i. In particular, there is no spin at all for a one-dimensional space and the situation is therefore trivial for the $1D$ CFT case. Since an analytical solution is known, we also decided to fix the spin values for the Ising model beforehand. This reduces the search space and makes the problem more similar to the $1D$ CFT, which is the main focus and innovative part of this thesis.

- A consequence of the unitarity of the models is not only that the squared OPE coefficients are both real and positive [RRTV08] but also that the scaling dimensions of the spectrum operators are bounded from below with a function of the spin and the dimension of the space $\Delta_i \geq f(s_i, d)$. For the Ising model we assume $\Delta_i \geq s_i$, while for the $1D$ CFT we will make the situation even simpler by applying previous results in the literature.

- We will assume that the squared OPE coefficients C_i^2 have values in the interval $[0, 1]$. This is true for the Ising model and is true for the first three squared OPE coefficients in the one-dimensional model as well [CGJP22a]

Finally, the environment representing our unknowns of the Ising $2D$ model is

$$(\Delta_1, \Delta_2, \ldots, \Delta_n, C_1^2, C_2^2, \ldots, C_n^2) \in [s_1, \Delta_{\max}] \times \ldots \times [s_n, \Delta_{\max}] \times [0, \infty)^n \tag{4.4}$$

Notice that we have considered a limited number of operators n. This value has to be defined in advance since it is the number of unknowns we will calculate being careful that the number of operators with $\Delta_i \leq \Delta_{\max}$ has to be greater or equal to n in order to ensure the search space is correct.

Recall that the crossing equation (4.1) involves the variables $z, \bar{z}$, which have continuous values, and that the equation must be satisfied by any value of such variables. The main approach to solve this issue is presented in [EvHS16] and consists in evaluating the (truncated) crossing equation only on a finite set of values for $z, \bar{z}$. This is an additional approximation we make on the problem and, as we will see later, may produce biased results.

For this to work the only requirement is to have enough points to determine all the parameters, which translates into having more evaluation points than unknowns $N_z > 2n$.

With this specification, we have reduced the original problem to a set of N_z equations that must all be satisfied with the CFT data $(\vec{\Delta}, \vec{C^2})$. These equations form a set of constraints, one for each value of $z_k, \bar{z}_k$

$$\begin{aligned}
&\sum_{i:\Delta_i \leq \Delta_{\max}} C_i^2 F_{\Delta_i}(z_1, \bar{z}_1) = 0 \\
&\qquad \ldots \\
&\sum_{i:\Delta_i \leq \Delta_{\max}} C_i^2 F_{\Delta_i}(z_k, \bar{z}_k) = 0 \\
&\qquad \ldots \\
&\sum_{i:\Delta_i \leq \Delta_{\max}} C_i^2 F_{\Delta_i}(z_{N_z}, \bar{z}_{N_z}) = 0
\end{aligned} \tag{4.5}$$

that we will indicate simply as $\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2}) = 0$.

At each step, the agent chooses an action that corresponds to selecting a couple of CFT parameters $A_t = (\Delta_j, C_j^2)$ where $1 \leq j \leq n$ and it does so by cycling through the values of j, starting from 1. In other terms, the agent chooses the parameters corresponding to operator j at every time step $t = kn + j$ with reminder j modulo n.

The state of the agent consists of the values last selected for the CFT data $S_t = (\vec{\Delta}, \vec{C^2})$ of which the last selected ones are those corresponding to the index $j = t \bmod n$.

The values of the crossing equations are $\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2})$ are then calculated and, since we made some approximations, we do not expect to solve the equations (4.5) exactly but we aim to minimize the numerical error and to have reminders as close as possible to zero. These reminders $\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2})$ form the observation O_t of the agent. In case the values are complex we use as observations just the real parts of the components of the vector.

The last quantity to be calculated from the environment is the reward R_t which will guide the agent's learning. Two different choices were available and used in literature [KPN22b], [KNPR23]. Consider the euclidean norm of the crossing equation vector $\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|$. We want this to be as close as possible to 0, while the reward is a quantity that we aim to maximize. To fit into this framework the two proposed solutions are:

- $R_t = -\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|$ as in [KPN22b].
- $R_t = \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|}$ as in [KNPR23].

Both options come from the same authors, with the work [KPN22b] being a refinement and extension of [KPN22b]. In this work both solutions were tried again, with the second leading to consistently better performance in terms of matching analytically known results for both models. As we will see, the reciprocal of the crossing equations norm is easier to integrate with additional constraints that we will discuss in the one-dimensional CFT. Therefore, the reward used to produce the final results is $R_t = \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|}$.

One final feature of the environment is deciding how an episode is terminated. The termination flag is triggered when the agent receives a reward that is greater than the previous best reward obtained. In this case, the best value is overwritten by the current reward and a new episode starts. In the algorithm, we will also see that an episode can be terminated also when the search is not improving for too many steps and the limit is defined with a threshold, called `faff_max`.

In table 4.1 we summarize the main concepts for the application of RL to conformal bootstrap as discussed.

RL	**CFT Bootstrap stochastic optimization**
Environment	Implementation of the conformal bootstrap equation
State	Current configuration of CFT data $(\vec{\Delta}, \vec{C^2})$
Action	Generate a couple of CFT data (Δ_i, C_i^2)
Observation	Evaluation of conformal bootstrap equation on current CFT data $\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2})$
Reward	$R = -\|\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2})\|$ or $R = \frac{1}{\|\vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2})\|}$
State transition	Deterministic, unknowns generated sequentially always in the same order
Step	Sample one action
Episode	Ends with `faff_max` number of steps of no improvement or when improvement wrt best reward
Policy, action value function, critic(s)	Neural Networks

Table 4.1: Summary of the main concepts of RL applied to conformal bootstrap

4.1.1 The algorithm

The RL algorithm used is the SAC described in section 3.4.2. This choice is due to the capabilities of SAC and the necessity of operating in continuous action and state spaces. The huge complexity of the problems we are dealing with actually requires a more complex application of the SAC algorithm involving re-initializations of the agent and a search space that gets increasingly narrowed around the best promising values.

Reinforcement learning tasks usually involve a number of parameters and values to be identified much lower than what we are working with here. In this case, the CFT data we aim to identify will be of around $10 - 25$ values leading to a very complex optimization problem also involving functions with high complexity.

When a run is initiated the agent moves in the environment continuously updating the memory buffer and optimizing all the neural networks involved in a common SAC algorithm: the value network, the two critic networks and the stochastic policy network. All these networks take as input the observation O_t which, as we have seen, is a function of the current state S_t and give as output respectively the evaluation of the current state, the soft Q-functions and the next action to be performed.

After some iterations, the reward stops improving. This is due to the fact that the agent is pursuing a policy that may not be optimal and is no more exploring all the state space. To overcome this problem the networks are re-initialized and the memory buffer is flushed leading to a completely new exploration by the agent. There is evidence that this approach leads to almost immediate improvement [KNPR23]. The parameters controlling the maximum number of iterations before this re-initialization is `faff_max`.

After some resets, there is no room for great improvements in the solution and it is necessary to increase the precision of parameters and the results by restricting the search window. The maximum number of re-initialized runs without improvements is set with the parameter called `pc_max`. Once this value is reached, the search windows are reduced by some percentage controlled by the `window_rate` parameter.

The new and narrowed search windows are centered around the CFT data values of the best reward obtained within the last search window. If the initial search is done in the whole space

$$[0, \Delta_{\max}]^n \times [0, 1]^n \tag{4.6}$$

After the first window reduction, the search space becomes

$$\begin{aligned} &\left[\Delta_1 - \frac{A}{2}, \Delta_1 + \frac{A}{2}\right] \times \ldots \times \left[\Delta_n - \frac{A}{2}, \Delta_1 + \frac{A}{2}\right] \times \\ &\left[C_1^2 - \frac{B}{2}, C_1^2 + \frac{B}{2}\right] \times \ldots \times \left[C_n^2 - \frac{B}{2}, C_n^2 + \frac{B}{2}\right] \end{aligned} \tag{4.7}$$

where $A = \texttt{window\_rate} \cdot \Delta_{\max}$ and $B = \texttt{window\_rate} \cdot 1$. At each subsequent quenching, the windows get reduced by additional factors `window_rate`. For each window size the same procedure above is repeated. The total number of window size reductions is defined by the parameter `max_windows_exp`.

At this point, the final results are the CFT data values that led to the maximum reward overall. This algorithm is presented in [KNPR23] and is fully automated in terms of re-initializations and window size decreases. The original implementation of [KPN22b] required more user inputs and initializations.

The full description of the algorithm can be found in algorithm 6.

Algorithm 6 Bootstrap Stochastic Optimization algorithm

Initialize parameters, best reward $R^* = 0$
while number of windows reductions less than `max_windows_exp` **do**
  while number of re-initializations less than `pc_max` **do**
    Initialize neural networks, agent and reset memory buffer.
    for Each time step t **do**
      Agent selects action (Δ_j, C_j^2) with $j = t \bmod n$.
      Update the state $S_t = (\vec{\Delta}, \vec{C^2})$.
      Calculate conformal blocks and crossing equations $O_t = \vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})$.
      Calculate reward $R_t = \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|}$.
      Agent receives observation O_t and reward R_t.
      Update memory buffer with the last transition.
      Update/learn parameters according to the main SAC algorithm
      if $R_t > R^*$ **then**
        Overwrite previous best reward R^* and agent restart episode, $t = 0$.
      end if
      if number of steps without improving reaches `faff_max` **then**
        Exit For loop and reinitialize
      end if
    end for
  end while
  Reduce search windows size by a factor of `window_rate` centered around the state correspondent to R^*.
end while

4.2 Additional remarks on the SAC implementation

Given the complexity of the optimization problem, it is in some cases useful to force some analytically or numerically known values in the search in order to reduce the overall number of variables in the search. To implement this two vectors of booleans are created:

- `guessing_run_list_deltas` for which the False values indicate fixed delta values.
- `guessing_run_list_opes` for which the False values indicate fixed squared OPE coefficient C^2 values.

Since the implementation of SAC uses the crossing equations as network inputs this operation, in some sense, does not break the differentiability of the network objective, although we are forcing some values into the states.

The initial sizes of the search windows are specified in the vectors of real components `guess_sizes_deltas` and `guess_sizes_opes`. Known and fixed values we want to force into the search will have a window size of 0, while unknown parameters have the corresponding size defined by our approximation $[0, \Delta_{\max}]$ or, in case of a non-zero spin s_i, $[s_i, \Delta_{\max}]$ with size $\Delta_i - s_i$ due to the constraint $\Delta_i \geq s_i$.

The lower bounds of the search are contained in the vectors `shifts_deltas` and `shifts_opes`. Within this framework the base value for these is 0 for unknown CFT

data and the forced value for known CFT data.

Lastly, two particular parameters control how the same spin operator values are treated:

- `same_spin_hierarchy`: if true, the operators with the same spin are ordered in such a way that, in each iteration and calculation, the scaling dimensions are increasing. This reordering does not break the differentiability of the neural networks' targets since, as noted previously, these are based on the rewards and the observations O_t which are independent of the order of the operators in the state S_t.

- `dyn_shift`: it is defined as the minimum difference between the scaling dimensions of operators with the same spin. At each time step we modify the current state S_t by imposing that, if operators $i, i+1$ have the same spin and $\Delta_{i+1} < \Delta_i + \texttt{dyn\_shift}$ the value of Δ_{i+t} is set to $\Delta_i + \texttt{dyn\_shift}$. This appears in [KNPR23], although in some cases we avoided using this feature since it forces a different value for a scaling dimension from the one suggested by the policy, while the above approach of reordering states does not modify the actual values given by the policy.

Values for $z, \bar{z}$

The fact that the crossing equations are evaluated in specific points of the complex plane $z, \bar{z}$ gives the additional problem of selecting these points optimally. Following [KNPR23], these points are chosen to have more stability and numerical precision in the calculations of the hypergeometric functions in the conformal blocks of both models. Common Python implementations of these functions give more accurate results in particular regions of the complex plane on which we focus.

In reality, a set of $N_z = 180$ couple of points is selected based on the latter fact and is kept fixed in all of the parallel runs. These values are given from the authors of [KNPR23].

Speed-up techiques

The calculation conformal blocks $F_{\Delta_i}(z_k, \bar{z}_k)$ are, by far, the slowest individual part of the whole process and a technique to significantly decrease computational time has been developed by sacrificing some numerical accuracy. First, notice how conformal blocks depend only on Δ_i and $z_k, \bar{z}_k$ with the latter being known and fixed. The main idea is to calculate the values of the conformal blocks beforehand for some values of Δ_i and approximate the needed calculations with the pre-determined values obtained:

- In case of forced Δ_i values there is no approximation needed: the final conformal blocks are calculated and saved for later use. The values are exact and there is no difference in terms of accuracy, while we save time in the agent's learning phases.

- In case of unknown values of Δ_i we select a grid of points in $[0, \Delta_{\max}]$ equally spaced with a difference between consecutive points as low as possible, selected to be 0.0005 or 0.00005. The values of the conformal blocks are precalculated for these specific values of Δ and, in the learning phase, instead of doing the exact calculation for Δ_i, we round this CFT data value to the closest $\Delta_{\text{grid}_i} \leq$

Δ_i in the point grid. The result will not be exact but this is just a minor approximation compared to the others we introduced above and will not affect the final calculation hugely. On the other hand, the almost 10 times speed increase in calculation enables us to do many more runs trying to find the optimal solution.

In the next sections, we present the two main models we will be applying the algorithm to.

4.3 2D Ising model

The 2D Ising model represents the interaction of magnetic spins in a plane and has been largely studied both from a statistical mechanics and a conformal field theory point of view. In the statistical mechanics' framework, we can imagine having a square lattice of N points with periodic conditions at the boundaries, effectively giving a torus structure. In each point i lies a magnetic spin σ_i taking values in $\{-1, 1\}$ representing magnets with two possible orientations. These lattice sites interact with each other with a coupling factor J_{ij} for a couple of points (i, j) and possibly with an external magnetic field for which the interaction with point j is given by h_j. The Hamiltonian governing the system is then given by

$$H = \exp\left(-\sum_i h_i\sigma_i - \sum_{i\neq j} J_{ij}\sigma_i\sigma_j\right) \tag{4.8}$$

Although seemingly easy, the $2D$ Ising model can show interactions between spins and unstable configurations such as the regular triangle, where every possible assignment of $\sigma_1, \sigma_2, \sigma_3$ leads to a non equilibrium. On the other hand, the Conformal field theory of the Ising $2D$ model has been widely studied and has an analytical solution we will use as a benchmark of the procedure.

The critical Ising $2D$ model is one of the simplest CFTs in the Euclidean space. It contains a total of three fields, the identity $\mathbb{I}$, the energy density operator $\epsilon(z, \bar{z})$ with conformal weights $\left(\frac{1}{2}, \frac{1}{2}\right)$ and the spin operator $\sigma(z, \bar{z})$ with conformal weights $\left(\frac{1}{16}, \frac{1}{16}\right)$. The four-point functions take the form

$$\begin{aligned}\langle\sigma(z_1,\bar{z}_1)\sigma(z_2,\bar{z}_2)\sigma(z_3,\bar{z}_3)\sigma(z_4,\bar{z}_4)\rangle &= \frac{1}{|z_{12}|^{\frac{1}{4}}|z_{34}|^{\frac{1}{4}}}G_\sigma(z,\bar{z})\\ \langle\epsilon(z_1,\bar{z}_1)\epsilon(z_2,\bar{z}_2)\epsilon(z_3,\bar{z}_3)\epsilon(z_4,\bar{z}_4)\rangle &= \frac{1}{|z_{12}|^2|z_{34}|^2}G_\epsilon(z,\bar{z})\end{aligned} \tag{4.9}$$

for which we know the true analytical solutions

$$\begin{aligned}G_\sigma(z,\bar{z}) &= \frac{1}{2[(1-z)(1-\bar{z})]^{\frac{1}{8}}}\left[\sqrt{(1+\sqrt{z})(1+\sqrt{\bar{z}})}+\sqrt{(1-\sqrt{z})(1-\sqrt{\bar{z}})}\right],\\ G_\epsilon(z,\bar{z}) &= \frac{1-z+z^2}{1-z}\frac{1-z\bar{z}+\bar{z}^2}{1-\bar{z}}\end{aligned} \tag{4.10}$$

Expanding the correlators in blocks one can obtain the spectrum and the OPE coefficients for this theory. In fact, this is used as a toy model to validate the approach and the algorithm presented above.

Consider for now the $\langle\sigma\sigma\sigma\sigma\rangle$ correlator. The OPE results in

$$G_\sigma(z,\bar{z}) = 1 + \sum_{h\geq\bar{h}} C^2_{h,\bar{h}} g^{(2D)}_{h,\bar{h}}(z,\bar{z}) \tag{4.11}$$

where $C^2_{h,\bar{h}} = C^2_{\sigma\sigma\phi_{h,\bar{h}}}$ is the usual squared OPE coefficient and the conformal weighs $h,\bar{h}$ are related to the spin and scaling dimension of the operator with the relations

$$\Delta = \frac{h+\bar{h}}{2}, \quad s = \frac{h-\bar{h}}{2}. \tag{4.12}$$

The conformal blocks can be written as

$$g^{(2D)}_{h,\bar{h}}(z,\bar{z}) = \frac{1}{1+\delta_{h,\bar{h}}}\left(g^{(1D)}_h(z)g^{(1D)}_{\bar{h}}(\bar{z}) + g^{(1D)}_h(\bar{z})g^{(1D)}_{\bar{h}}(z)\right) \tag{4.13}$$

where we have introduced the $1D$ conformal blocks. To understand their structure we give the following definition.

Definition 4.1. The **hypergeometric function** ${}_2F_1(a,b,c;z)$ is the solution to the Euler's hypergeometric differential equation

$$z(1-z)\frac{d^2f}{dz^2} + [c-(a+b+1)z]\frac{df}{dz} - abf = 0 \tag{4.14}$$

and can be expressed as a series

$${}_2F_1(a,b,c;z) = \sum_{n=0}^{\infty} \frac{(a)_n(b)_n}{(c)_n}\frac{z^n}{n!} \tag{4.15}$$

where we have introduced the symbol $(q)_n$ as

$$(q)_n = \begin{cases} 1, & n=0, \\ q(q+1)\cdots(q+n-1), & n>0 \end{cases} \tag{4.16}$$

The $1D$ conformal block is then given by

$$g^{(1D)}_h(z) = z^h\,{}_2F_1(h,h,2h;z) \tag{4.17}$$

Recall that the main object we want to solve is the Conformal Bootstrap equation (2.140) which in this case takes the form

$$\begin{aligned} &((1-z)(1-\bar{z}))^{\frac{1}{8}} - (z\bar{z})^{\frac{1}{8}} + \\ &+ \sum_{h,\bar{h}} C^2_{h,\bar{h}}\left[((1-z)(1-\bar{z}))^{\frac{1}{8}}\, g^{(2D)}_{h,\bar{h}}(z,\bar{z}) - (z\bar{z})^{\frac{1}{8}} g^{(2D)}_{h,\bar{h}}(1-z,1-\bar{z})\right] = 0 \end{aligned} \tag{4.18}$$

The $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator has a very similar structure and, with the same steps, the conformal bootstrap equation can be found to be

$$\begin{aligned} &((1-z)(1-\bar{z})) - (z\bar{z}) + \\ &+ \sum_{h,\bar{h}} C^2_{h,\bar{h}}\left[((1-z)(1-\bar{z}))\, g^{(2D)}_{h,\bar{h}}(z,\bar{z}) - (z\bar{z}) g^{(2D)}_{h,\bar{h}}(1-z,1-\bar{z})\right] = 0 \end{aligned} \tag{4.19}$$

Our final unknowns of the problem are the scaling dimensions $\Delta = \frac{h+\bar{h}}{2}$ and the squared OPE coefficients $C^2_{h,\bar{h}}$, since we simplified the original problem by fixing the spin values which, as a remark, are all even in this theory. The procedure to determine the set of unknowns is the following:

- We choose a value for $\Delta_{\max}$, in this case 10, as the cutoff. This also limits the possible spin values of the operators to be $s_i \leq \Delta_i$.
- For the values of the admissible spins, we select all operators such that $\Delta_i \leq \Delta_{\max}$. This is the final spectrum of operators for which the scaling dimension and squared OPE coefficient has to be found by the algorithm. Recall that for the Ising $2D$ model conformal invariance implies that the values for the spins s_i are always even [RRTV08].

4.4 1D defect CFT on the Half-BPS Wilson line

While the study of the $2D$ Ising model is mainly a review of existing results in the literature, we now turn to the main focus of this thesis, which contains our original results: the study of the $1D$ defect CFT defined by a straight $\frac{1}{2}$-BPS Wilson line in $4D$ $\mathcal{N}=4$ super Yang-Mills theory [Mal98, DK06b, GRT17].

The $1D$ defect CFT we consider lives in a 4 dimensional space but is indeed 1 dimensional. It is invariant under the $1D$ conformal group $SO(1,2)$ but not under the 4 dimensional conformal group. The theory has a fundamental parameter g: for small values $g \to 0$ of it we talk about weak coupling and, vice versa, for higher values we talk about strong coupling. Both the scaling dimension and the OPE coefficients, as well as other quantities, depend on g. There exist analytical tools that let us study with perturbation theory the CFT data only for small g [CGJP22a], [GGJ20] or large g [FM21], [LMM18], while for intermediate values of the coupling constant only numerical methods are available. The objective of this work is to study how the CFT data varies as a function of the parameter g and fill in the gaps left.

The most important operator in the theory is a scalar of dimension $\Delta = 1$, which we call $\phi^1_\perp$. Following [CGJP22b], [CGJP22a], its 4-point correlator can be written as

$$\langle \phi^1_\perp(x_1)\phi^1_\perp(x_2)\phi^1_\perp(x_3)\phi^1_\perp(x_4)\rangle = \frac{G(x)}{x_{12}^2 x_{34}^2}. \tag{4.20}$$

Notice that since we are in a one-dimensional setting there exists only one cross ratio $z = \bar{z} = x = \frac{x_{12}x_{34}}{x_{13}x_{24}}$. As before, we work with values of x in the complex plane $\mathbb{C}$ except for the lines on the real axis $(-\infty, 0]$ and $[1, \infty)$. The crossing equation can be written as

$$x^2 G(1-x) - (1-x)^2 G(x) = 0 \tag{4.21}$$

A physical symmetry known as supersymmetry poses constraints on $G(x)$ [LMM18] which, following [CGJP22a], can be solved by

$$G(x) = \mathbb{F}x^2 + \left(\frac{2}{x} - 1\right) f(x) - (x^2 - x + 1) f'(x) \tag{4.22}$$

where $\mathbb{F} = \mathbb{F}(g)$ is a constant and the function $f(x)$ satisfies the crossing equation

$$x^2 f(1-x) + (1-x)^2 f(x) = 0 \tag{4.23}$$

We now consider the OPE for the function $f(x)$, written as

$$f(x) = F_{\mathcal{I}}(x) + C^2_{BPS} F_{\mathcal{B}_2}(x) + \sum_{n=1}^{\infty} C_n^2 F_{\Delta_n}(x) \tag{4.24}$$

with the conformal blocks given by the formulas

$$\begin{aligned} F_{\mathcal{I}}(x) &= x \\ F_{\mathcal{B}_2}(x) &= x - x\ {}_2F_1(1,2,4;x) \\ F_{\Delta}(x) &= \frac{x^{\Delta+1}}{1-\Delta}\ {}_2F_1(\Delta+1,\Delta+2,2\Delta+4;x) \end{aligned} \tag{4.25}$$

Recall that, in the one-dimensional case, there is no spin and operators are parametrized only by their scaling dimension Δ_n and have an associated squared OPE coefficient C_n^2, where $n \in \mathbb{N}$ is an index.

Although no analytical results are available in the literature, previous studies showed that it is possible to obtain precise numerical values for the scaling dimensions of the operators in the OPE, as well as bounds on some of the OPE coefficients. In particular, the numerical values of the scaling dimensions Δ_n for $1 \leq n \leq 10$ can be obtained from a series expansion as a function of the coupling g [GGJ20], which can be used to produce their numerical values with arbitrary precision. This is quite helpful for us since we can fix those values in the algorithm beforehand and simplify the search for the remaining CFT data. The availability of these results is also the main reason we experiment on the $2D$ Ising model by fixing an increasing number of scaling dimensions to further validate the approach.

The values of Δ_n for the first 10 operators can be found in [CGJP22a, GGJ20]. From these articles, we also inherit the ordering and indexing of these operators from weak coupling. In particular, the 10 operators considered are the ones with the lowest scaling dimension for $g = 0$, ordering them by their Δ values sto that $\Delta_n < \Delta_{n+1}$ at $g = 0$. We stress that this ordering on operators is kept the same throughout all values of the coupling constant, even if for large g it may happen that $\Delta_n > \Delta_{n+1}$ (see also figure 4.1 below).

Figure 4.1, taken from [CGJP22a], shows the scaling dimensions of the first 10 operators as a function of g. We remark on the fact that Δ values for different operators can approach the same numbers.

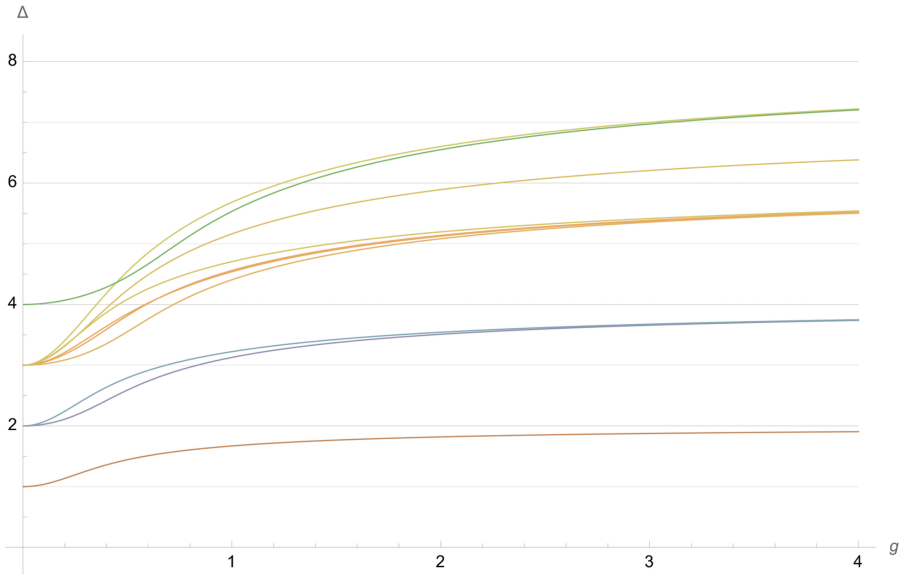


Figure 4.1: Values for the scaling dimension Δ_n for the first 10 operators of the OPE as a function of the coupling dimension g, ref. [CGJP22a]

In order to express the constants used in equations (4.22) and (4.25) we define a particular class of functions

Definition 4.2. The **(modified) Bessel functions of the first kind** I_α is one the solutions of the following differential equation, also called **Bessel's equation**

$$x^2\frac{d^2f}{dx^2} + x\frac{df}{dx} + (x^2 - \alpha^2) = 0 \tag{4.26}$$

where $\alpha \in \mathbb{C}$. In particular, the Bessel functions of the first kind satisfy particular constraints on the boundary and can be written as a series expansion

$$I_\alpha(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha} \tag{4.27}$$

With this in mind, we can finally define the constants $\mathbb{F}(g)$ and C^2_{BPS} as

$$\begin{aligned} \mathbb{F}(g) &= \frac{3I_1(4\pi g)((2\pi^2 g^2+1)I_1(4\pi g) - 2g\pi I_0(4\pi g))}{2g^2\pi^2 I_2(4\pi g)^2} \\ C^2_{BPS}(g) &= \mathbb{F}(g) - 1 \end{aligned} \tag{4.28}$$

4.4.1 Integral constraints and the bounds on OPE coefficients

The $1D$ CFT we are studying has two additional constraints we can impose on its 4-point function, as shown in [CGJP22a]. To be more specific, they are integral constraints that involve integrals of the conformal blocks. To write them we need to first define two constants (dependent on the coupling g), the Bremsstrahlung function $\mathbb{B}(g)$ given by [CGJP22a]

$$\mathbb{B}(g) = \frac{g}{\pi}\frac{I_2(4\pi g)}{I_1(4\pi g)} \tag{4.29}$$

and the curvature $\mathbb{C}(g)$. The latter has a complex definition that requires the introduction of many concepts. For simplicity, we instead write the Taylor expansion of if for weak coupling $g \to 0$ and strong coupling $g \to \infty$ being respectively [CGJP22a]

$$\begin{aligned} \mathbb{C}(g) = 4g^4 &- \left(24\zeta_3 + \frac{16\pi^2}{3}\right)g^6 + \left(\frac{64\pi^2\zeta_3}{3} + 360\zeta_5 + \frac{64\pi^4}{9}\right)g^8 \\ &- \left(\frac{112\pi^4\zeta_3}{5} + 272\pi^2\zeta_5 + 4816\zeta_7 + \frac{416\pi^6}{45}\right)g^10 \\ &+ \left(\frac{3488\pi^6\zeta_3}{135} + \frac{2192\pi^4\zeta_5}{9} + \frac{9184\pi^2\zeta_7}{3} + 63504\zeta_9 + \frac{176\pi^8}{15}\right)g^12 + O(g^14) \end{aligned} \tag{4.30}$$

and

$$\begin{aligned} \mathbb{C}(g) = &\frac{(2\pi^2-3)g}{6\pi^3} + \frac{-24\zeta_3 + 5 - 4\pi^2}{31\pi^4} + \frac{11+2\pi^2}{256\pi^5 g} + \frac{96\zeta_3 + 75 + 8\pi^2}{4096\pi^6 g^2} \\ &+ \frac{3(408\zeta_3 - 240\zeta_5 + 213 + 14\pi^2)}{65536\pi^7 g^3} + \frac{3(315\zeta_3 - 240\zeta_5 + 149 + 6\pi^2)}{65536\pi^8 g^4} + O\left(\frac{1}{g^5}\right) \end{aligned} \tag{4.31}$$

where ζ_n is the Riemann zeta function $\zeta_n = \sum_{k=1}^{\infty}\frac{1}{k^n}$.

We are now ready to define the integral constraints [CGJP22a] which will play a fundamental role in the discussion about OPE coefficients and in the experimental results:

$$\begin{aligned} &\text{Constraint 1:} \quad \int_0^1 \delta G(x) \frac{1+\log x}{x^2} \mathrm{d}x = \frac{3\mathbb{C}(g) - \mathbb{B}(g)}{8\mathbb{B}(g)^2} \\ &\text{Constraint 2:} \quad \int_0^1 \frac{\delta f(x)}{x} \mathrm{d}x = \frac{\mathbb{C}(g)}{4\mathbb{B}(g)^2} + \mathbb{F}(g) - 3 \end{aligned} \tag{4.32}$$

where $\delta G(x) = G(x) - G^{(0)}_{\text{weak}}(x), \delta f(x) = f(x) - f^{(0)}_{\text{weak}}(x)$, and $G^{(0)}_{\text{weak}}(x), f^{(0)}_{\text{weak}}(x)$ are the zero coupling values

$$\begin{aligned} G^{(0)}_{\text{weak}}(x) &= \frac{2(x-1)x+1}{(x-1)^2} \\ f^{(0)}_{\text{weak}}(x) &= 2x + \frac{x}{x-1} \end{aligned} \tag{4.33}$$

To include these constraints into our algorithm, we write them in a simpler way involving the CFT data, using the results in [CGJP22a] which are obtained by applying the OPE expansion and the crossing equation to (4.32). Define the integral functions as

$$\begin{aligned} \text{Int}_1[F_{\Delta_n}] &= -\int_0^{\frac{1}{2}} (x - 1 - x^2) \frac{F_{\Delta_n}}{x^2} \partial_x \log(x(1-x)) \mathrm{d}x \\ \text{Int}_2[F_{\Delta_n}] &= \int_0^{\frac{1}{2}} \frac{F_{\Delta_n}(2x-1)}{x^2} \mathrm{d}x \end{aligned} \tag{4.34}$$

and the numerical constraints as

$$\begin{aligned} \text{RHS}_1 &= \frac{\mathbb{B}(g) - 3\mathbb{C}(g)}{8\mathbb{B}(g)^2} + \left(7\log 2 - \frac{41}{8}\right)(\mathbb{F}(g) - 1) + \log 2 \\ \text{RHS}_2 &= \frac{1 - \mathbb{F}(g)}{6} + (2 - \mathbb{F}(g))\log 2 + 1 - \frac{\mathbb{C}(g)}{4\mathbb{B}(g)^2} \end{aligned} \tag{4.35}$$

With these equations in mind the integral constraints in (4.32) become

$$\begin{aligned} &\text{Constraint 1:} \quad \sum_n C_n^2 \text{Int}_1[F_{\Delta_n}] + \text{RHS}_1 = 0, \\ &\text{Constraint 2:} \quad \sum_n C_n^2 \text{Int}_2[F_{\Delta_n}] + \text{RHS}_2 = 0, \end{aligned} \tag{4.36}$$

These integral constraints were used to limit the possible values of the OPE coefficients in [CGJP22a]. In particular, the authors showed a considerable decrease in width for the region of admissible values of the squared OPE coefficients, especially C_2^2 and C_3^2. Figure 4.2, taken from [CGJP22a], shows the numerical bounds for the first 3 squared OPE coefficients without implementing any integral constraint into the algorithm used by the authors of the work.

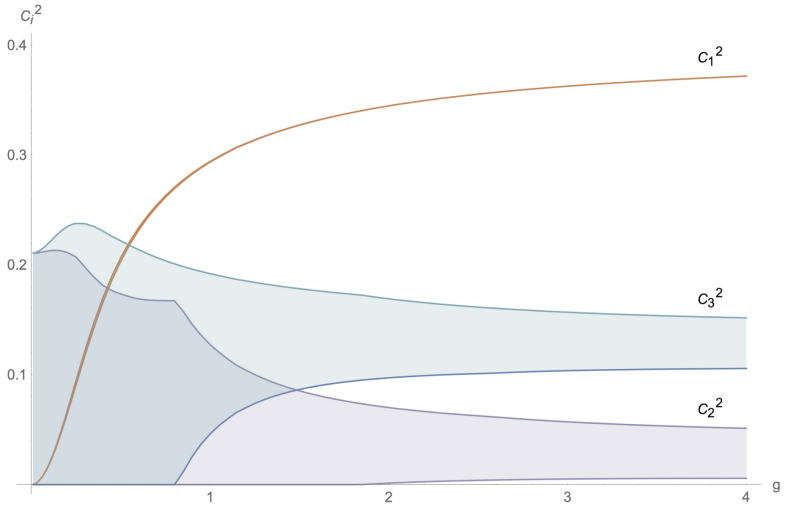


Figure 4.2: Bound regions for the first 3 squared OPE coefficients as a function of g. Ref. [CGJP22a]

From this first figure we can notice that, while the bounds for C_1^2 are small for all values of the coupling constant, the relative errors on C_2^2 and C_3^2 are high and increase as $g \to 0$. By including one of the integral constraints into the procedure the results dramatically improve [CGJP22a]:

- The width of the bounds on C_1^2 decreases by at least a factor of 10 for all coupling constants and even more for larger values of g, with the first integral constraints giving the best results.
- Looking at C_2^2 and C_3^2, for both coefficients and for either constraint applied, the region shrinks by a factor of 2 starting at $g = 0.3$ and a factor of 9 at strong coupling.

Figure 4.3 shows graphically the improvement we described.

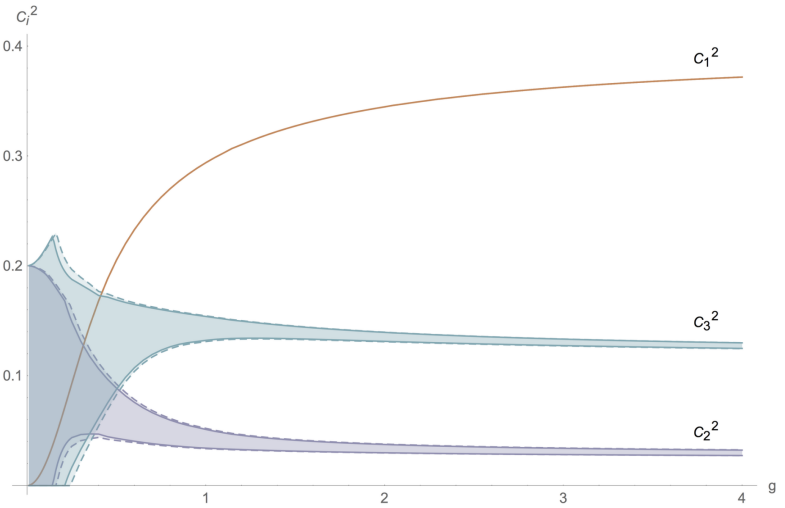


Figure 4.3: Bound regions for the first 3 squared OPE coefficients as a function of g, first integral constraint applied (hard lines) or second integral constraint applied (dashed lines). Ref. [CGJP22a]

Although better, the relative error on C_2^2 and C_3^2 is still not satisfactory and the precision is still lost at small coupling. The best results are obtained by applying both integral constraints into the framework [CGJP22a]:

- For C_1^2 the width decreases by at least 10^3 on all coupling values.
- For C_2^2 and C_2^2 the bound reduces monotonically with the coupling, for $g = 0.3$ by at least a factor of 10, which becomes 10^2 at $g \sim 1.5$ and almost 200 at strong coupling.

Figure 4.4, taken from [CGJP22a] as well, shows this clearly.

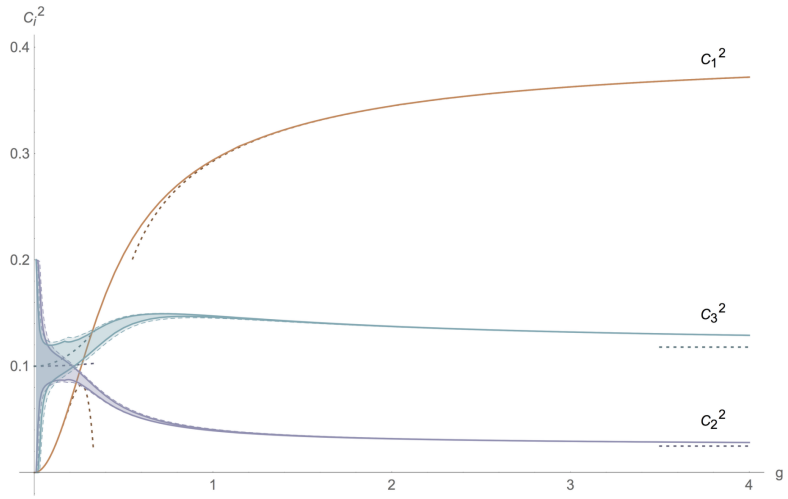


Figure 4.4: Bound regions for the first 3 squared OPE coefficients as a function of g, both integral constraints applied. Ref. [CGJP22a]

The values for the second and third OPE coefficients are now very precise for g large enough and, as we further discuss later, can be added as an input to simplify the problem even more. At small coupling, on the other hand, the precision is still lost, although remarkably better, and we may use the bounds to validate the approach to find new CFT data.

Finally, the same approach could be applied to further operators but the results are less precise and only a non-trivial upper bound is found [CGJP22a].

4.4.2 Final remarks on the implementation

Before going into the experimental setup and results, there are some final remarks to be made for the implementation of the 1D defect CFT we study. Our objective is to give numerical estimations of the unknown CFT data for the first 10 operators and study them as a function of the coupling constant g.

No analytical results are available for this model although, as we discussed earlier, we know the scaling dimensions Δ_n for $1 \leq n \leq 10$ with arbitrary high precision [CGJP22a], [GGJ20]. This is still a good advantage compared to no information at all, which is the case for most theories. These values are given as input to the algorithm to reduce the dimensionality of the problem and to produce more accurate results.

If we want to simplify the problem even more, we could take advantage of the bounds on C_n^2 for $n = 1, 2, 3$ found in [CGJP22a]. The relative error on the first squared OPE coefficient is very small for all the values of g between 0 and 4, with a precision of minimum 10^{-7}. Hence, in some experiments, we may give a value in this range as an input to the algorithm. Unfortunately, the bounds for C_2^2 and C_3^2 are wide for $g < 1$ and lose precision for small coupling $g \leq 0.5$, even with two integral constraints applied. This can be clearly seen from figure 4.4. For large coupling, on the other hand, the allowed region is small enough to fix values beforehand in the algorithm to reduce complexity even more.

For these squared OPE coefficients, we only have an upper and a lower bound and hence we need to decide which numerical value we give as input. Given the discussion above, we use this additional information only when the allowed region is small enough, with the input value being fixed to the mean of the upper and the lower bound. In reality, when the bounds are close enough, any value within the range could be considered, as there would not be a significant difference in the results.

The last issue we need to tackle is how to include the integral constraints into our framework without further increasing the complexity of the procedure. Recalling equation (4.5), we can reformulate the entire task with the following $N_z + 2$ equations

$$\begin{aligned} \vec{\boldsymbol{E}}(\vec{\Delta}, \vec{C^2}) &= 0 \\ \mathbb{I}_1 = \sum_n C_n^2 \text{Int}_1[F_{\Delta_n}] + \text{RHS}_1 &= 0 \\ \mathbb{I}_2 = \sum_n C_n^2 \text{Int}_2[F_{\Delta_n}] + \text{RHS}_2 &= 0 \end{aligned} \tag{4.37}$$

Note that all of them involve the same CFT data and our goal is to have the left-hand sides of these equations as close as possible to 0. One could consider the same reward as before $R_t = \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|}$ while $\mathbb{I}_1$ and $\mathbb{I}_2$ as pure constraints in a constrained problem. Unfortunately, the soft Actor-Critic algorithm does not have an efficient way to include such constraints without considerable effort.

Our solution is to include the integral constraints into the reward in the same way we are using the crossing equation. The structure and the data needed are the same so this procedure is indeed straightforward. We will try two possible forms of rewards:

$$\begin{aligned} R_1 &= \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|} + w_1 \frac{1}{|\mathbb{I}_1|} + w_2 \frac{1}{|\mathbb{I}_2|} \\ R_2 &= \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\| + w_1 |\mathbb{I}_1| + w_2 |\mathbb{I}_2|} \end{aligned} \tag{4.38}$$

where w_1 and w_2 are weights to be determined and kept fixed afterward.

Both possibilities come from the same idea: the closer our equations are to zero, the greater the reward the agent gets, and the better the results. The main difference lies in how we treat each constraint compared to the others:

- In the first case, each constraint contribution to the reward is completely independent from the others. If we suppose without loss of generality that the optimal weights are $w_1 = w_2 = 1$, a total reward of order 1000 can be obtained with $\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\| \simeq 10^{-3}, \mathbb{I}_1 \simeq 1, \mathbb{I}_2 \simeq 1$ as well as $\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\| \simeq 1, \mathbb{I}_1 \simeq 10^{-3}, \mathbb{I}_2 \simeq 1$ and $\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\| \simeq 1, \mathbb{I}_1 \simeq 1, \mathbb{I}_2 \simeq 10^{-3}$.

- In the second case, the contributions of the three constraints are much more related. In fact, to have a reward of order 1000 all three quantities must be close to zero, at least each one being $\simeq 10^{-3}$. This guarantees that if an optimal policy is found, all our requests are greatly satisfied.

We will see that R_2 produces results that are more satisfactory and controllable but we will analyze both for a complete discussion.

4.4.3 Previous approaches in literature

We are now going to review recent numerical approaches in the literature for this model based on concepts such as integrability [GKP98,BAA+11,DKN+19,Lip94,FK95, MZ03] and the formalism of Quantum Spectral Curve (QSC) [GKLV14, GKLV15] to put further constraints on correlators. These are the main ideas behind the numerical exploration of the spectrum of the first 10 operators mentioned above.

The set of bounds for the first three squared OPE coefficients was obtained in [CGJP22a, CGJP22b] using methodologies that were modifications of the original numerical bootstrap [RRTV08, KPSD14] we presented in section 2.5.1. They search for a functional of the form

$$\alpha[F(x)] = \sum_{n=0}^{N_{\text{det}}/2} A_n \partial_x^{2n} F(x)|_{x=\frac{1}{2}} \tag{4.39}$$

which, applied to the conformal bootstrap equation gives

$$\sum_n C_n^2 \alpha[\mathcal{G}_{\Delta_n}(x)] + \alpha[\mathcal{G}_{\text{simple}}(g,x)] = 0 \tag{4.40}$$

where $\mathcal{G}_{\text{simple}}(g,x)$ and $\mathcal{G}_{\Delta_n}(x)$ contain the crossing equations respectively for the first two conformal blocks in (4.24) and the conformal blocks of the first 10 operators of the spectrum.

Equation (4.40) can be rewritten as

$$\sum_n C_n^2 \alpha[\vec{A}\cdot\vec{V}_{\Delta_n}] + \alpha[\vec{A}\cdot\vec{V}_{\text{simple}}] = 0 \tag{4.41}$$

where $\vec{A} = (A_0, A_1, \dots, A_{N_{\text{der}}/2})$ and $\vec{V}_{\Delta_n}, \vec{V}_{\text{simple}}$ are respectively the vectors of derivatives of $\mathcal{G}_{\Delta_n}, \mathcal{G}_{\text{simple}}$ at $x = \frac{1}{2}$. The positivity condition of $\vec{A}\cdot\vec{V}_\Delta \geq 0$ for $\Delta \geq \Delta_*$ implies the condition at the base of the following algorithms:

$$\sum_{\Delta_n < \Delta_*} C_n^2 \alpha[\vec{A}\cdot\vec{V}_{\Delta_n}] + \alpha[\vec{A}\cdot\vec{V}_{\text{simple}}] \leq 0 \tag{4.42}$$

For example, fixing $\Delta_* = \Delta_2$ we can search for $\vec{A}^{\text{up}}$ such that $(\vec{A}^{\text{up}}\cdot\vec{V}_{\Delta_1}) = 1$ and $(\vec{A}^{\text{up}}\cdot\vec{V}_{\text{simple}})$ is maximal, obtaining un upper bound $C_1^2 \leq -(\vec{A}^{\text{up}}\cdot\vec{V}_{\text{simple}})$. A lower bound can be found analogously changing the signs. This is now formulated as a problem of Numerical Conformal Bootstrap that can be solved as before by using polynomial approximation and semi-definite programming.

The algorithm can be further improved by including information on the rest of the spectrum $\Delta_1, \dots, \Delta_N$. The positivity condition becomes $(\vec{A}\cdot\vec{V}_\Delta) \geq 0$, $\forall \Delta \geq \Delta_N$

together with the discrete constraints $(\vec{A}\cdot\vec{V}_{\Delta_n})\geq 0$ for $n=1,\dots,N-1$. We are now dealing with a bigger space of functionals since $(\vec{A}\cdot\vec{V}_{\Delta})$ can be negative in the intervals between the values of the spectrum.

For each $m=1,\dots,N-1$ we can now impose the above positivity conditions for $n\neq m$ and look for $\vec{A}^{\text{up},m},\vec{A}^{\text{low},m}$ such that

- $(\vec{A}^{\text{up},m}\cdot\vec{V}_{\Delta_m})=1$.
- $(\vec{A}^{\text{up},m}\cdot\vec{V}_{\text{simple}})$ is maximal.
- $(\vec{A}^{\text{low},m}\cdot\vec{V}_{\Delta_m})=-1$.
- $(\vec{A}^{\text{low},m}\cdot\vec{V}_{\text{simple}})$ is maximal.

obtaining the bounds

$$(\vec{A}^{\text{low},m}\cdot\vec{V}_{\text{simple}})\leq C_m^2\leq -(\vec{A}^{\text{up},m}\cdot\vec{V}_{\text{simple}}) \tag{4.43}$$

The algorithm can be improved again by adding information on previously found bounds when increasing m, although the best improvement in the results comes from introducing into the framework the integral constraints (4.32). The integration is indeed very easy since equation (4.36) has a very similar form to (4.41). The final equation, from which the procedure is the same as the algorithms described above, is

$$\begin{aligned}\sum_n C_n^2\left[\sum_{k=0}^{N_{\text{der}}/2} b_k\partial_x^{2k}\mathcal{G}_{\Delta_n}|_{x=\frac{1}{2}}+b_{-1}\text{Int}_1[F_{\Delta_n}]+b_{-2}\text{Int}_2[F_{\Delta_n}]\right]\\ +\sum_{k=0}^{N_{\text{der}}/2} b_k\partial_x^{2n}\mathcal{G}_{\text{simple}}|_{x=\frac{1}{2}}+b_{-1}\text{RHS}_1+b_{-2}\text{RHS}_2=0\end{aligned} \tag{4.44}$$

The images shown before highlight the importance of such integrated constraints on the first three squared OPE coefficients, with the bounds reducing by a factor of 30 for weak coupling values and 80 for strong couplings. Unfortunately, as stated in [CGJP22a], the bounds obtained on $C_i^2, i=4,\dots,10$ are not precise and satisfactory. We will see as, in some cases, our approach can identify values for these coefficients with some degree of precision, particularly for couplings that are neither too big nor too small.

Chapter 5

Results

In this Chapter, we present the experimental results of our study described in chapter 4. Ultimately we are interested in the $1D$ defect CFT defined on the $\frac{1}{2}$-BPS Wilson line in the $4D$ $N = 4$ super Yang-Mills theory, where no prior application of RL exists in literature and no complete solutions to the crossing equation ((4.23) and (4.24)) is available. As discussed, we focus on finding the squared OPE coefficients C_i^2 and use the numerical values of the scaling dimensions Δ_i as input.

We start by describing the experimental setup and the preliminary results to find the parameters that perform the best. Then, we follow the literature on the RL approach to CFT [KPN22a], [KPN22b], using the $2D$ Ising as a toy model in section 5.2. Initially, the search is kept free with unknowns given by the whole CFT data $(\vec{\Delta}, \vec{C^2})$ and then we fix an increasing number of scaling dimensions in order to see if results improve when the values of Δ are given as input.

Finally, we show the results from the experiments on the one-dimensional CFT. The first objective is to determine the best formula for the reward between the two available in (4.38). Then, we analyze the squared OPE coefficient found as a function of the coupling constant g, making a distinction between weak coupling $g < 1$ and strong coupling $g \geq 1$.

5.1 Experimental Setup

All the following experiments were conducted using a modified version of the code shared by the authors of [KNPR23] available on the BootSTOP GitHub page[1]. The particular code used in this work can be found on the BootSTOP CFT GitHub page[2] by Alessandro Trenta. The main structure has been maintained to have consistency for the modified SAC algorithm. The key difference stands in the implementation of our models of interest, the Ising 2D model and the Half-BPS Wilson line defect CFT, as well as the integration with the additional integral constraints (4.36) of the latter model. The code was run on Python 3.8.13, while the neural networks were implemented using the library Pytorch [PGM+19] at version 1.12.0 which handled also their optimization.

To obtain statistically relevant results, for each experiment we set up from 500 to 2000 parallel runs depending on the experiment, with each one taking between 2

[1] https://github.com/vniarchos/BootSTOP

[2] https://github.com/AlexThirty/BootSTOP_CFT

and 3 days. Each run has a different random seed so that we can assume them to be independent to calculate the mean and standard deviation. This was possible thanks to a cluster at the University of Pisa, consisting of 16 nodes with 96 CPUs each. The scaling of the parallel runs on the cluster was handled by the Ray library [MNW+18] (version 1.13 and 2.2) and SLURM.

While the majority of the parameters described in chapter 4 were optimized using a grid search, we chose to fix some of them to values found by the original authors to avoid having a very large search space. In particular, the neural networks used in Soft Actor-Critic are all Fully Connected Neural Networks with 2 hidden layers and an output layer of 1 unit, except for the actor network which has 2 outputs as it needs to calculate the mean and variance of the distribution generating the action $\mu_\phi(S_t), \sigma_\phi(S_t)$.

The other parameters defining the networks can be found in table 5.1.

Parameter	Value
Layer 1 units	256
Layer 2 units	256
Activation function layer 1	ReLU
Activation function layer 2	ReLU
Batch size	256

Table 5.1: Values for Neural Networks specific parameters. These are fixed for all the experiments in this thesis.

5.2 The benchmark model: Ising 2D

The Ising $2D$ model described in section 4.3 has been widely studied in literature and has a well-known analytical solution of the CFT data $(\vec{\Delta}, \vec{C^2})$, both for the $\langle\sigma\sigma\sigma\sigma\rangle$ and the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator. Ising $2D$ has also been used as a toy model in [KPN22b], with a method similar but prior to the Bootstrap Stochastic Optimization with Soft Actor-Critic, showing the power of this framework. We reproduced the results of the original article, this time with the approach described in [KNPR23] and in the chapter 4, which features the automation in the window reduction, not present in the first version.

In all the following discussions and experiments the maximum value for Δ was set to 10.5. We also used the second speedup technique described in the previous chapter by precalculating the values for the conformal blocks for conformal dimensions of the form $\Delta = k \times 0.0005$, until 10.5. The grid of Δ values has then a width of 0.0005.

Before the main experiments, a grid search was performed on some parameters of the algorithm to find the values that maximize the reward obtained by the agent. In this phase, we analyze the results in a completely agnostic way from the underlying model: our aim is to purely maximize the reward at some time step in the process, without checking that the CFT data found $(\vec{\Delta}, \vec{C^2})$ is close to the known analytical solution. In fact, maximizing the reward is the same as minimizing the norm of the crossing equation vector $\vec{E}(\vec{\Delta}, \vec{C^2})$ which, at least theoretically, corresponds to finding the CFT data describing the model.

Recall that operators in the Ising $2D$ model have an (always even [RRTV08]) spin associated. This requires us to set the value of the `dyn_shift` parameter, which is the minimum distance between scaling dimensions of operators with same spin. Although

this parameter was initially included in the grid search, we decided not to use it to apply less forcing on CFT data and let the agent more freely explore the environment. On the other hand `same_spin_hierarchy` was applied since it involves just a visual reordering of the scaling dimension of operators with the same spin. The window reduction factor, `window_rate`, was set to 0.7 following [KNPR23] as well as to not shrink the search too early, leaving some chance to explore more.

In table 5.2 we present the parameters involved in the search with the corresponding values tried and the final choice.

Parameter	**Set of values**	**Final choice**
`faff_max`	$\{100, 500, 1000, 5000, 10000\}$	10000
`pc_max`	$\{5, 10, 15, 20, 25\}$	10
`max_windows_exp`	$\{5, 10.15, 20, 25\}$	25
τ	$\{0.05, 0.005, 0.0005\}$	0.0005
α	$\{0.0001, 0.0005, 0.001, 0.005\}$	0.0005
β	$\{0.0001, 0.0005, 0.001, 0.005\}$	0.0005
reward scale (α^{-1})	$\{0.001, 0.01, 0.1, 1, 10\}$	0.001

Table 5.2: Grid search summary for parameters of the algorithm: $2D$ Ising model

The term τ is the exponential smoothing parameter of the value network, α is the learning rate for the actor network and β is the learning rate for the rest of the networks. The discount rate γ is set to the value of 0.99 to improve the search of basins of attraction. Since the crossing equation is continuous in the CFT data, a high discount value should help push the agent toward finding the right path to the solution. The reward scale corresponds to the inverse of the α factor in the maximum entropy objective for SAC 3.44.

The final values found by the grid search can be used to analyze the overall behavior of the algorithm. In particular

- The value of `faff_max` is indeed very important. It regulates how many time steps an episode can last without improvement and, in general, the higher the better. This is due to the fact that the agent is free to explore for much more time before the networks are initialized again.

- `pc_max` was set to 10 since higher values didn't improve much the reward. This parameter corresponds to the maximum number of initializations before reducing the search and thus adding precision. After 10 iterations the reward rarely improved, basically wasting computational time.

- `max_windows_exp` is the number of window reductions and therefore is linked to the final numerical precision of the CFT data. Intuitively, the higher this value the better the final result and this was also proven experimentally. The final window size, with a window reduction rate of 0.7 as above, is 0.0001 which is also less than the approximation we are using for the grid of Δ for the precalculated conformal blocks.

- τ, α, β did not have much influence on the final rewards. Therefore we used the same values by the original authors.

- The reward scale seemed not to have a high influence on the final results. Since a higher reward scale is linked to a lower α in equation 3.44 which implies less exploration, we chose to take a low reward scale in order to favor a less conservative agent.

The above parameters will be used in all the experiments involving the Ising $2D$ model.

5.2.1 Free search experiments

In this section we try to reproduce the results of [KPN22b] using the new and improved methodology by [KNPR23]. The objective is to test the framework by trying to find the analytical values defining the two correlators of the $2D$ Ising model.

$\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator

Table 5.3 contains the theoretical values and the experimental CFT data found for the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator with $\Delta_{\text{max}} = 10.5$ on the single run with the best reward. The experimental CFT data shown in table 5.3 corresponds to the values that the agent was using as CFT at the single time step it obtained his maximum reward possible.

Spin	**Analytic** Δ	**RL** Δ	**Analytic** C	**RL** C
0	4	2.93519	1	1.27600
0	8	5.20440	0.01	0.23713
2	2	2.36300	1	0.43162
2	6	5.27179	0.01	0.17654
2	10	7.08422	7.9365×10^{-4}	4.78596×10^{-3}
4	4	5.32145	0.1	0.0428243
4	8	7.88979	7.9365×10^{-3}	4.84263×10^{-3}
6	6	6.57830	7.9365×10^{-3}	1.29584×10^{-3}
6	10	8.12713	5.8275×10^{-4}	2.18920×10^{-3}
8	8	9.60932	5.8275×10^{-4}	2.56530×10^{-4}
10	10	10.14738	4.1135×10^{-5}	3.01555×10^{-5}

Table 5.3: Theoretical and experimental values for the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator in the Ising $2D$ model. The table contains only the results for the run with the best overall reward.

As we can see the results are not satisfactory and different from the theoretical ones. Furthermore, putting the theoretical CFT data in the crossing equations $\vec{E}(\vec{\Delta}_{\text{true}}, \vec{C^2}_{\text{true}})$ on the selected points in the complex plane gives a reward of 2586.0985, while the reward obtained using the above values is 3814.1957. This is a very interesting fact since the experimental reward is higher than the one obtained with the theoretical true values. This could be explained by two facts:

- We are considering a truncated version of the crossing equation (4.1). The theoretical values for this correlator may not solve this equation exactly, while there may exist other values of Δ and C, close to the one we obtained, which solve the truncated equation but not the original one.

- We are not solving the crossing equation on the whole complex plane (except where it is not defined), but we are just finding the best fit for the 180 points we selected to evaluate it.

This is an important remark to make and we will later see that fixing theoretical values in the algorithm can help the model to find the correct values instead of a particular solution unrelated to the analytical one.

In order to have a more complete understanding of the stability of the method and the statistical relevance of the results found, in table 5.4 we include the averages of the CFT data for the 25 runs that obtained the best rewards out of all the 450 parallel tries.

Spin	**Analytic** Δ	**RL** Δ	**Analytic** C	**RL** C
0	4	2.98961	1	1.01143
0	8	6.09123	0.01	0.37989
2	2	2.40974	1	0.59418
2	6	4.39658	0.01	9.83291×10^{-2}
2	10	6.93158	7.9365×10^{-4}	7.09948×10^{-2}
4	4	5.17865	0.1	5.34511×10^{-2}
4	8	8.31381	7.9365×10^{-3}	6.16930×10^{-3}
6	6	6.85276	7.9365×10^{-3}	3.71636×10^{-3}
6	10	8.70140	5.8275×10^{-4}	1.12237×10^{-3}
8	8	9.05906	5.8275×10^{-4}	3.40398×10^{-4}
10	10	10.30735	4.1135×10^{-5}	4.69906×10^{-5}

Table 5.4: Theoretical and experimental values for the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator in the Ising $2D$ model. The table contains the average values for the 25 runs with the best rewards.

We also included a graphical representation of the values for the CFT data found by the best 25 runs, their experimental means and the theoretical values. As we can see in figures 5.1 and 5.2 the results are very spread out, meaning that they lack consistency with a high relative error for all the CFT data. This can also be acknowledged by the fact that the average reward for the runs considered is 1990.8672 with a standard deviation of 519.1846: results are inconsistent between the best-performing runs.

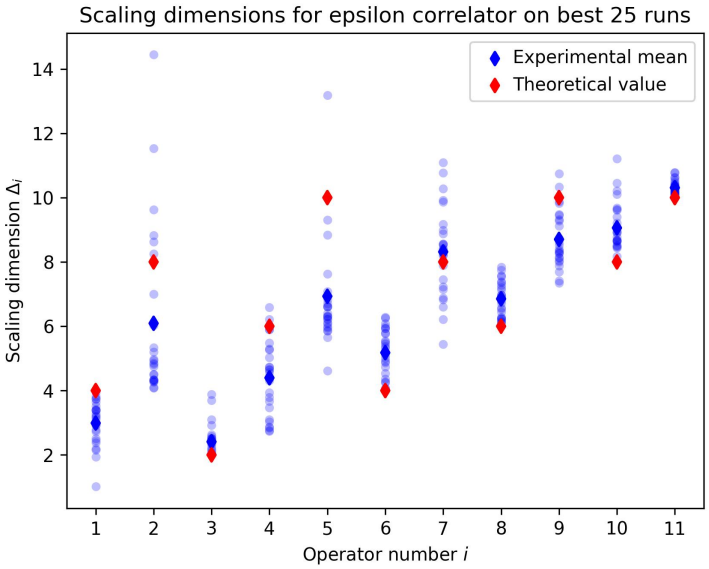


Figure 5.1: Results for the scaling dimensions Δ_i of the operators in the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator of the Ising $2D$ model. Red points represent the theoretical values and blue points represent experimental values for the 25 runs with the best reward, together with their average.

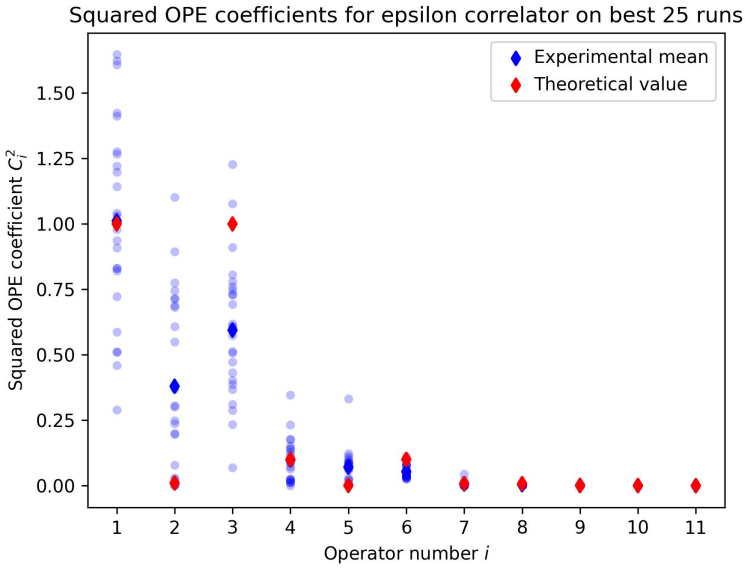


Figure 5.2: Results for the squared OPE coefficients C_i^2 of the operators in the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator of the Ising $2D$ model. Red points represent the theoretical values and blue points represent experimental values for the 25 runs with the best reward, together with their average.

We remark that this kind of problem is indeed very complex. We are trying to solve an equation with 22 variables and non-linear relations, where the usual Reinforcement Learning tasks involve from 1 to 3 outputs at a time. Our next objective will be to see how fixing some known values can help the search both by reducing the number of unknowns and the complexity of the problem in general.

We performed the same experiment for the $\langle\sigma\sigma\sigma\sigma\rangle$ correlator obtaining worse results. In this case, the problem is even more complex with a total of 32 unknowns. We refer to the appendix A.1 for the specific results for this correlator.

5.2.2 Constrained search experiments

For now, we only consider the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator.

In this particular set of experiments, we aim to see how performance improves when giving as input to the algorithm some (and possibly all) the known scaling dimension values for the operators. We introduce this approach since, in the $1D$ CFT we will analyze later in this chapter, we fixed the known Δ values to obtain statistically relevant results on the unknown OPE coefficient values.

In theory, we should expect that the more values we fix, the closer the remaining values are to the analytical ones. This is due to the two following facts:

- The search space becomes smaller and smaller as we fix values. When all Δ's are fixed, only the 11 OPE coefficient of the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ remain unknown.

- The more values we fix, the simpler the landscape of the crossing equation becomes with respect to the remaining unknowns. Intuitively, there will be less relevant minima of the norm of the crossing equations and, for the agent, it will be easier to find the optimal solution.

In this experiment, we began by fixing the scaling dimension values starting from the ones that have a more significant impact on the results, namely the smallest Δ's. We can think that the relevance or magnitude of an operator in the crossing equation as being proportional to $e^{-\Delta}$ due to equation (4.2). To distinguish between operators with the same scaling dimension, we started with the ones with a lower spin value. Recall that for a single spin value, there can only be just one operator with a particular Δ value, meaning that the order of fixing scaling dimensions is unique.

Table 5.5 summarizes the conformal blocks in the model and the way we ordered them for fixing their values. In total, we fix from 0 to 11 values of scaling dimensions,

Operator number	Spin	Δ	Sequential order
1	0	4	2
2	0	8	6
3	2	2	1
4	2	6	4
5	2	10	7
6	4	4	3
7	4	8	9
8	6	6	5
9	6	10	10
10	8	8	8
11	10	10	11

Table 5.5: Sequential order for giving as input the values of the operators in the $2D$ Ising model for the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator

that is 12 different experiments. The number of individual runs per experiment was 50 for a total of 600 parallel runs.

We start our analysis by looking at the rewards. Figure 5.3 shows the individual reward from the best performing run, colored in green, and the average reward for the top 10 runs per experiment, colored in blue, with a lighter area representing a 1 standard deviation zone for those runs.

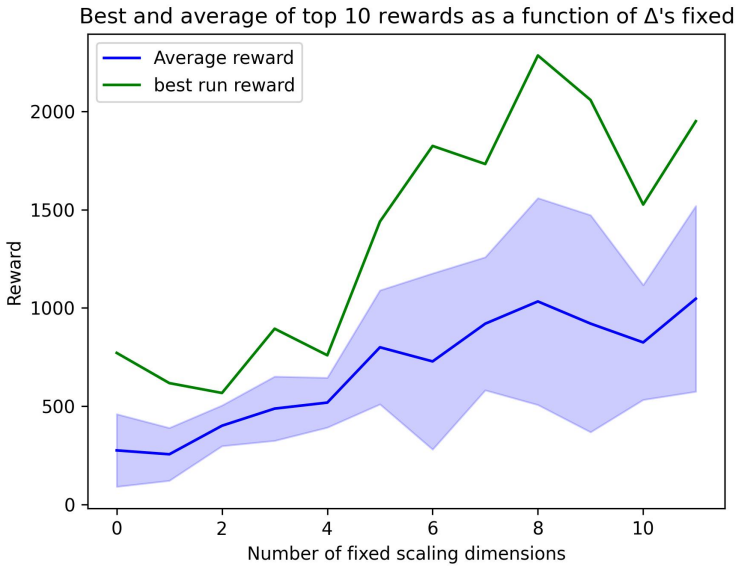


Figure 5.3: Best and averaged values for the reward in the constrained search experiment as a function of the number of Δ's given as input. $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator in the Ising $2D$ model. Only the 10 runs with the best reward are considered.

We can see that, as expected, the reward generally improves as we fix more scaling dimension values. The number of attempts per experiment is not very high and therefore a large uncertainty on the average reward or a non-monotonical behavior is expected.

What we are really interested in is the relative error on the OPE coefficients with respect to their theoretical value, in relation to the number of Δ values fixed, as in the $1D$ CFT model the formers will be the only unknowns. In Figure 5.4 we can see how the relative error of the experimental OPE coefficient for each operator changes as more and more scaling dimensions are fixed. For this plot, we only considered the best run for each experiment.

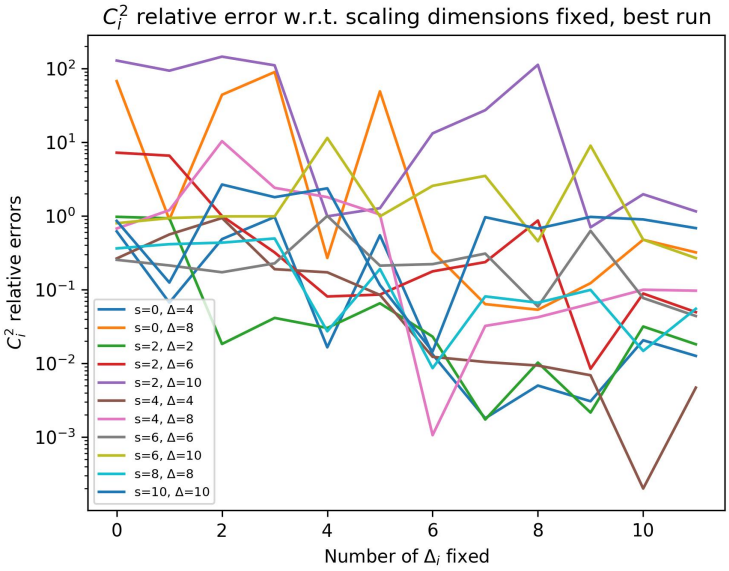


Figure 5.4: Relative error on the squared OPE coefficients C_i^2 as a function of the number of scaling dimensions Δ_i given as input. $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator in the $2D$ Ising model. Only the best run for each case is considered.

Let us make some comments on the results:

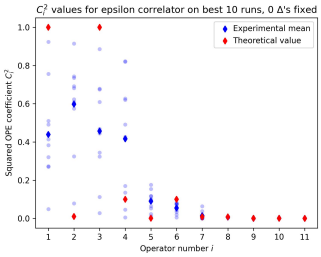
- It can be clearly seen that results improve almost monotonically as the number of fixed scaling dimensions increases. The results are therefore promising and encourage us to perform this experiment on the $1D$ CFT model, where every Δ is known numerically and used as input.

- It is not only the relative error for each operator that gets better but the stability of the obtained values improves as well, as it can be seen from figures 5.5. The standard deviation in the squared OPE coefficient found is smaller when more scaling dimensions are fixed. This suggests that simplifying the problem by reducing the number of unknowns also reduces the complexity of the landscape of the function, leading to more stable results from the SAC agent.

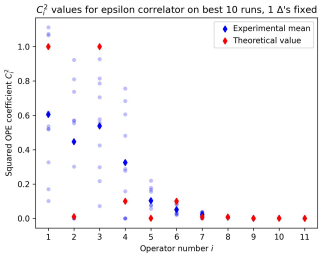
- Notice how in figure 5.4 there are 3 operators with a big relative error when only a few Δ's are fixed, followed by a significant drop in this error. These operators are in fact some of the highest dimensional ones with spectrums of $s = 0, \Delta = 8$, $s = 2, \Delta = 10$ and $s = 6, \Delta = 10$. As we expect, the highest dimensional

operators close to the cutoff are the hardest ones to find since they might be also compensating for the approximation made.

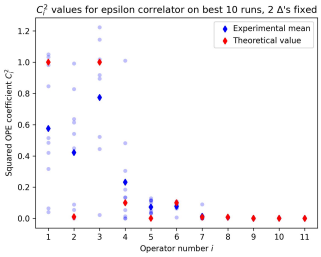
As a final remark, the same analysis and experiments were performed on the $\langle\sigma\sigma\sigma\sigma\rangle$ correlator. In this case the increased complexity of the problem, starting from 32 unknowns to 16 in total, made the search o for the CFT data very hard and the results are unsatisfactory. However, even for this correlator, the relative error on the squared OPE coefficients decreases as we give more Δ's as input. For the specific results we refer to the appendix A.1.



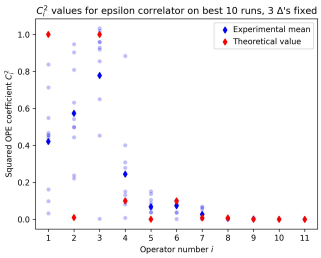
(a) 0 scaling dimension fixed



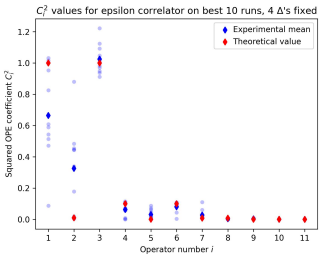
(b) 1 scaling dimension fixed



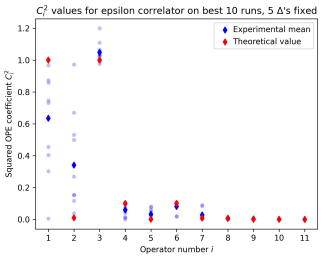
(c) 2 scaling dimension fixed



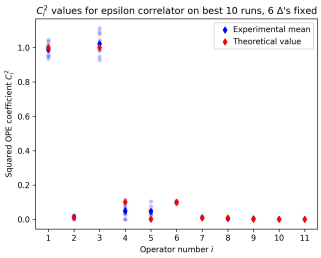
(d) 3 scaling dimension fixed



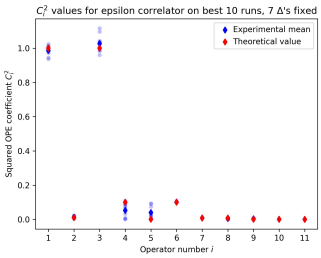
(e) 4 scaling dimension fixed



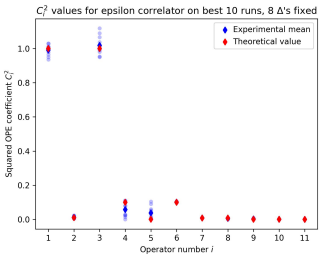
(f) 5 scaling dimension fixed



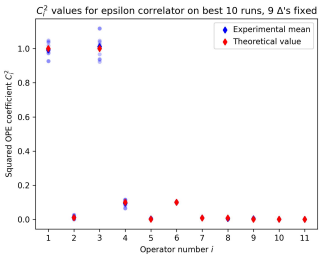
(g) 6 scaling dimension fixed



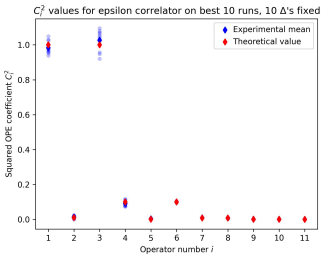
(h) 7 scaling dimension fixed



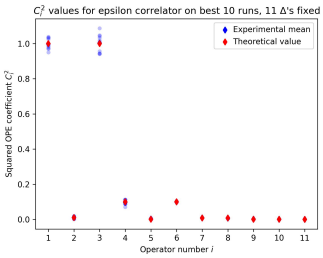
(i) 8 scaling dimension fixed



(j) 9 scaling dimension fixed



(k) 10 scaling dimension fixed



(l) 11 scaling dimension fixed

Figure 5.5: Results on the squared OPE coefficients C_i^2 for the experiment with an increasing number of scaling dimensions Δ_i fixed. $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator in the $2D$ Ising model. Red points represent the theoretical values while blue points represent the experimental values for the best 10 runs, together with their averages.

5.3 Half-BPS Wilson line defect CFT

In this section we will present the experiments we performed for the $1D$ defect CFT defined on the $\frac{1}{2}$-BPS Wilson line in the $4D$ $N = 4$ super Yang-Mills theory. Previous researches [GGJ20] were able to evaluate numerically with high precision the values of the scaling dimensions for the first 10 operators, as well as some bounds on the first three squared OPE coefficients [CGJP22a]. With in mind the results obtained in the last section, we focus on finding the values of the squared OPE coefficients for operators from the fourth to the tenth, while also making a few comments on higher-dimensional operators.

5.3.1 Initial search and the Role of Integral Constraints

Throughout this section, every experiment will have a coupling dimension of $g = 1$ since it is a good compromise between weak and strong coupling, with the bounds on the first three squared OPE coefficients being sufficiently tight as well.

The first step in our research is understanding the role and the power of the integral constraints in (4.32). First, we need to perform experiments in order to choose the formula for the reward from the two possibilities presented before:

$$\begin{aligned} R_1 &= \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\|} + w_1 \frac{1}{|\mathbb{I}_1|} + w_2 \frac{1}{|\mathbb{I}_2|} \\ R_2 &= \frac{1}{\|\vec{\boldsymbol{E}}_t(\vec{\Delta}, \vec{C^2})\| + w_1|\mathbb{I}_1| + w_2|\mathbb{I}_2|} \end{aligned} \tag{5.1}$$

Using the same parameters as for the Ising $2D$ model, we performed two grid searches on the weights w_1 and w_2 for both R_1 and R_2. These weights play a fundamental role since they determine the relative influence between the three components of the reward: the norm of the crossing equations vector and the integral constraints.

An additional problem in this analysis is that the rewards cannot be directly compared purely as numbers, since the weights influence both the magnitude and which term will be dominant in the sum for R_1 or in the denominator of R_1. To have a benchmark of the performance of a particular combination of weights we considered the values of the crossing equations norm and the integral constraint, as well as the bounds for the first three squared OPE coefficients.

In particular, we set as unknown all the squared OPE coefficients, while fixing the scaling dimensions for the first ten operators. Let C_i^2 be the squared OPE coefficient for the i-th operator, with $i = 1, 2, 3$. We know that for $g = 1$ we should have $m_i \leq C_i^2 \leq M_i$ where m_i, M_i were provided by [CGJP22b]. Suppose the location of the best reward for a run has a coefficient indicated by $\hat{C}_i$. We calculate the error of the estimate $\hat{C}_i$ as

$$\mathrm{Err}(\hat{C}_i) = \begin{cases} \frac{|m_i - \hat{C}_i|}{m_i} & \text{if } \hat{C}_i < m_i \\ 0 \text{ if } m_i \leq \hat{C}_i \leq M_i \\ \frac{|M_i - \hat{C}_i|}{M_i} & \text{if } \hat{C}_i > M_i \end{cases} \tag{5.2}$$

that is the relative error with respect to the closest bound if the estimate is outside the range. This will serve as additional information to the goodness of a combination of weights, since it represents the objective we are pursuing: finding the squared OPE coefficients.

We performed a search on the values of the weights for both reward formulas. For R_1 we considered $w_1, w_2 = 10^{-10}, 10^{-9}, \ldots, 10^{-3}$ for a total of 64 combinations, while for R_2 we tries the values corresponding to $w_1, w_2 = 10^{-1}, 10^{0}, \ldots, 10^{6}$ for a total of, again, 64 combinations. In each case, 10 runs for each combination were performed, resulting in 640 parallel runs for both grid searches.

In table 5.6 we present the most relevant combinations we found for the two possible reward formulas with the metrics we considered above. Recall that, for both the integral constraints and the crossing equations norm, the closer they are to 0, the better.

	R_1	R_2
w_1	10^{-7}	10^{4}
w_2	10^{-10}	10^{5}
Crossing equations norm	1.56739×10^{-3}	2.60363×10^{-3}
Integral constraint 1	5.82228×10^{-4}	2.37865×10^{-8}
Integral constraint 2	9.09057×10^{-5}	2.05983×10^{-9}
Err$(\hat{C}_1)$	1.5552×10^{-3}	2.12069×10^{-5}
Err$(\hat{C}_2)$	0.13186	0.001183
Err$(\hat{C}_3)$	0.01571	0.0
Reward	638.00146	377.6412

Table 5.6: Metrics for the best-performing weights for both rewards

As we can see, the results seem generally better for the second kind of reward R_2. Let us now analyze these results more in-depth:

- The errors on the first three squared OPE coefficients are much lower for R_2, we may therefore have a better chance to find relevant results for the other C_i^2 we do not have any information on.

- The integral constraints are far closer to 0 for R_2, by about 4 order of magnitude.

- The crossing equations norm is closer to 0 for R_1, but in this case by just a factor of 2. This observation, with the previous one, still leads to a better general performance by R_2.

- As noted in the previous chapter, R_2 forces all the 3 components in the reward to have the same order of magnitude to improve the overall reward. In the experiments involving R_1, we noted that sometimes and for some values of the weights the agent finds very quickly a policy that minimizes one of the integral components. This leads to less further improvement in the crossing equations norm which is still the most important component among the three.

Therefore, we chose to perform the rest of the experiments using the reward formula of R_2.

Since the reward values can be significantly different from the ones in the Ising $2D$ model, we performed an additional grid search on the reward scale and the learning rates for the neural networks, using the values we just identified for the weights. The results can be found in table 5.7

Parameter	Values tried	Final value
α	$\{0.0001, 0.0005, 0.001, 0.005\}$	0.005
β	$\{0.0001, 0.0005, 0.001, 0.005\}$	0.005
reward scale (α^{-1})	$\{0.001, 0.01, 0.1, 1, 10\}$	10

Table 5.7: Grid search summary for parameters of the algorithm: $1D$ CFT model

In this case, the quantity to maximize was the overall reward and the selected values were the ones performing best, although just slightly and without relevant differences. All the subsequent experiments will then be performed using these parameters as our final choice.

Before we show and comment on the experimental results for the weak and strong coupling cases, which are treated separately, let us make some remarks on the rewards that were obtained. Figure 5.6 shows the best reward from the experiments, both from the single top-performing run as well as the average of the top 25 runs.

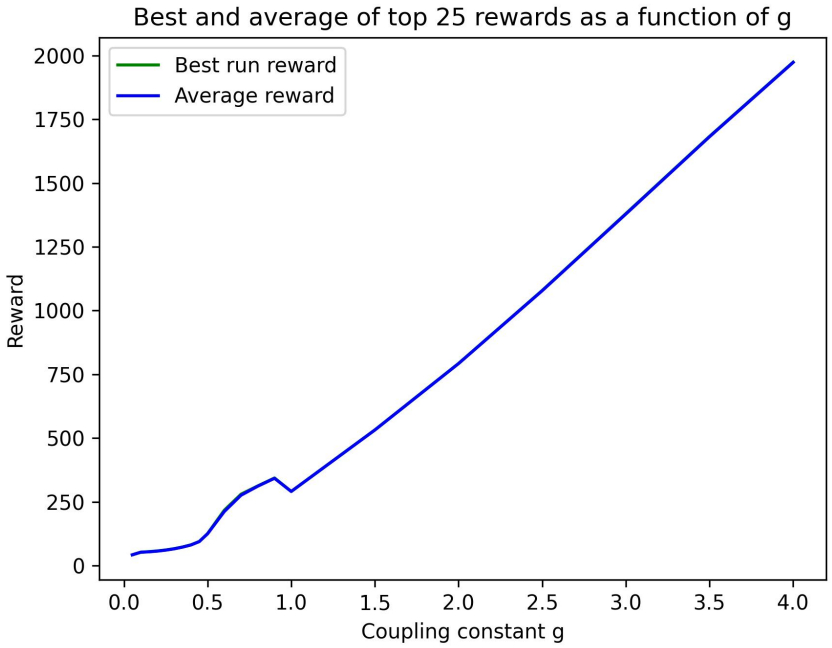


Figure 5.6: Experimental results on the values of the rewards for the one-dimensional CFT model as a function of the coupling constant g. For each value of g we considered the best 10 performing runs.

First, note the stability of the results obtained: the error band around the averages with 1 standard deviation width is invisible and the average almost overlaps with the reward from the best run. Therefore, we expect good precision in the C_i^2 estimates, at least for lower dimensional operators. As we will see, this is not always the case, due to the similarity of the scaling dimensions of some operators.

Second, the rewards increase monotonically as g grows, which means that the solutions found by the algorithm to the crossing equations and the integral constraints are more accurate at strong coupling. Therefore, we expect more accurate results for large values of g. The small bump at $g = 1$ could be explained by the change in the

setting we apply at this value: for $g < 1$ we only give C_1^2 as input while, for $g \geq 1$, we fix both C_1^2, C_2^2 and C_3^2.

Finally, figure 5.7 shows the reward as a function of the number of timesteps of exploration by the agent, with the coupling constant fixed to 1 as reference. In particular, the blue lines and the light blue regions represent respectively the average and standard deviation of the reward for the 10 runs with higher overall performance. The rewards are averaged over 100 timesteps. Green vertical lines represent the approximated timesteps at which the neural networks are reset (the exact values are different within each run), while red vertical lines correspond to the approximate timesteps where the search window is reduced.

From the plot it is clear that the approach is working and the agent is learning correctly how to achieve greater rewards, with the help of the search window reductions as well.

5.3.2 Experiments at weak coupling

We start from the weak coupling case, in particular for values of $g \leq \frac{1}{2}$. For these values of the coupling constant, the available bounds for the second and third squared OPE coefficients are wide, as already discussed in section 4.4. The precision is not high enough to select a value inside the range provided by [CGJP22b] as input for the algorithm, as we do for $g \geq 1$ in section 5.3.3. Without a very good approximation for C_2^2 and C_3^2, results for further OPE coefficients would be biased toward any decision made or, in general, not relevant.

For the first squared OPE coefficient C_1^2, on the other hand, we can fix this value without major concerns for all values of $g \geq \frac{1}{100}$, since the bounds provided by [CGJP22a] have a width at least 8 orders of magnitude smaller than the lower bound. The differences in values within these windows are mostly negligible for our approach and algorithm.

In these experiments, we search for the values of the squared OPE coefficients C_i^2 for $2 \leq i \leq 10$ with a particular focus on C_2^2 and C_3^2, in order to see if our results are in accordance with the ones obtained by previous works with other methodologies. The selected coupling constants are

$$g = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \tag{5.3}$$

and, for each value, we ran 500 parallel runs.

Figures 5.8 and 5.9 contain the experimental results for the 25 runs with the best reward on C_2^2 and C_3^2. Colored in blue, we plot the individual values of the coefficient for each run as well as their average value, while the theoretical bounds from [CGJP22a] are represented by the green regions. In appendix A.2 we also give the numerical values for our estimates and the upper and lower bounds (refer to table A.1).

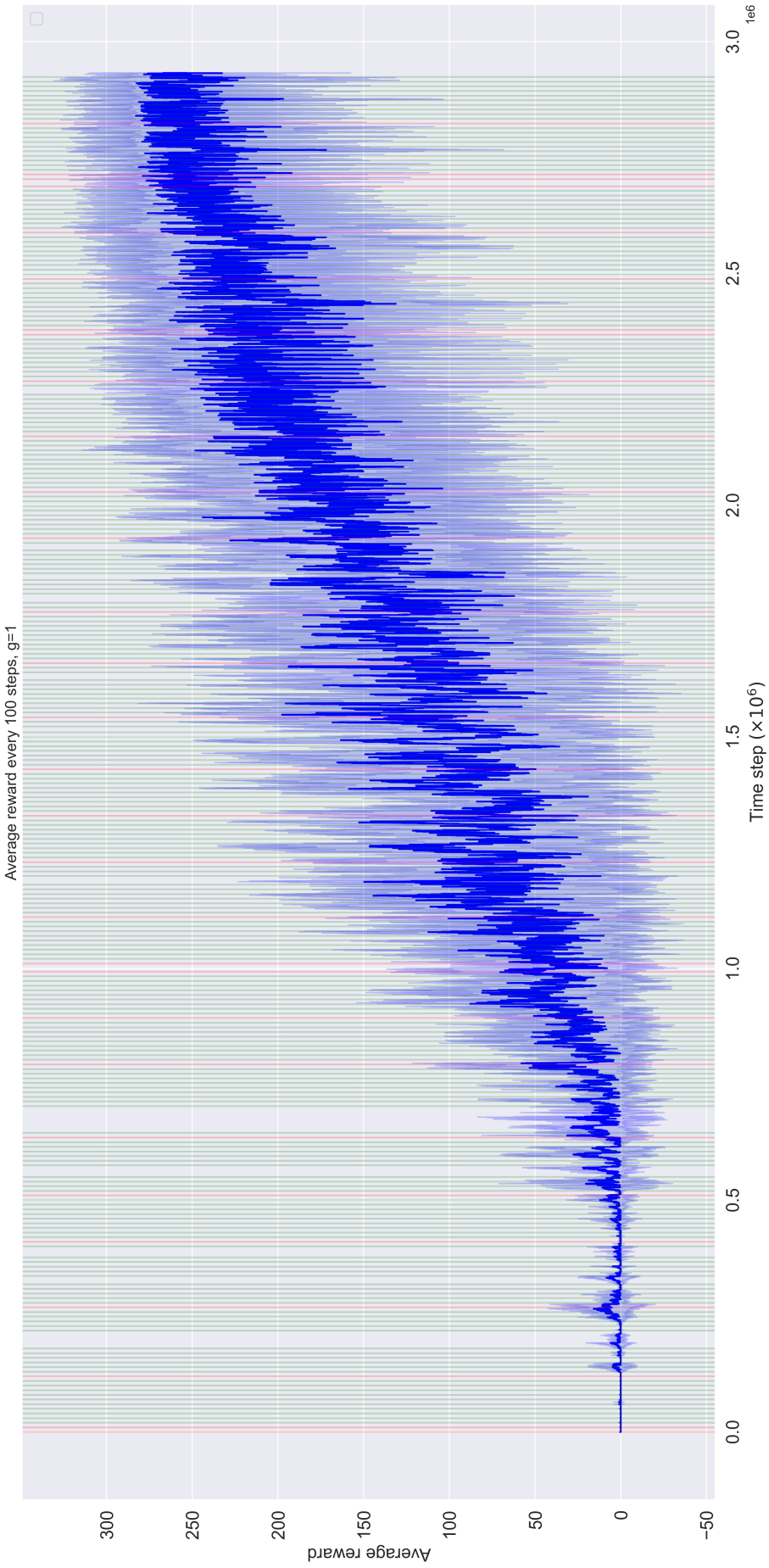


Figure 5.7: Reward as a function of the number of timesteps of exploration in the one-dimensional model with coupling constant $g = 1$. Blue points represent the averages for the top 10 runs, with light blue regions of height 1 standard deviation. Green vertical lines represent the approximate time steps at which the neural networks are reset, with red lines representing the window reductions.

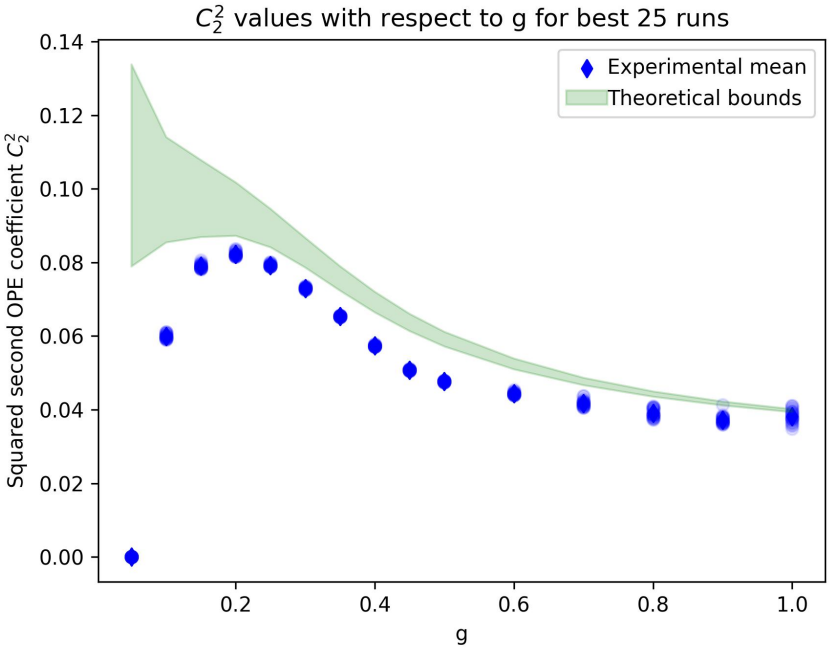


Figure 5.8: Theoretical bounds [CGJP22a] and experimental values for C_2^2 in the one dimensional CFT model with $g \leq 1$. Only the top 25 runs are considered.

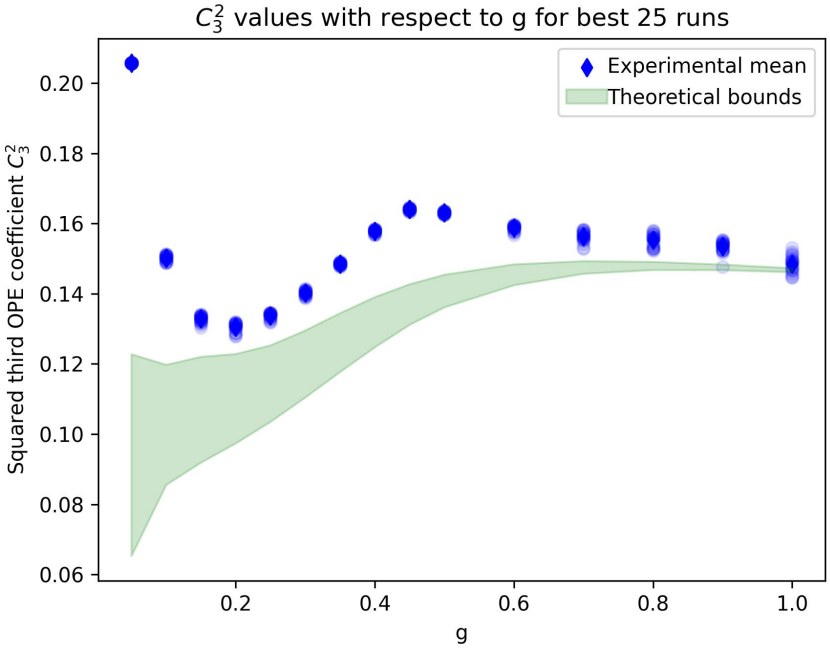


Figure 5.9: Theoretical bounds [CGJP22a] and experimental values for C_3^2 in the one dimensional CFT model with $g \leq 1$. Only the top 25 runs are considered.

As we can see from figures 5.8 and 5.9, the experimental estimates for C_2^2 and C_3^2 are outside of the expected range, although the available windows are wide. The precision is also very high, meaning that the algorithm always tends to find a minimum that it believes contains the solution to the original problem. If we instead consider the sum

$C_2^2 + C_3^2$, the results found are more coherent with the theoretical bounds, as we can see from figure 5.10.

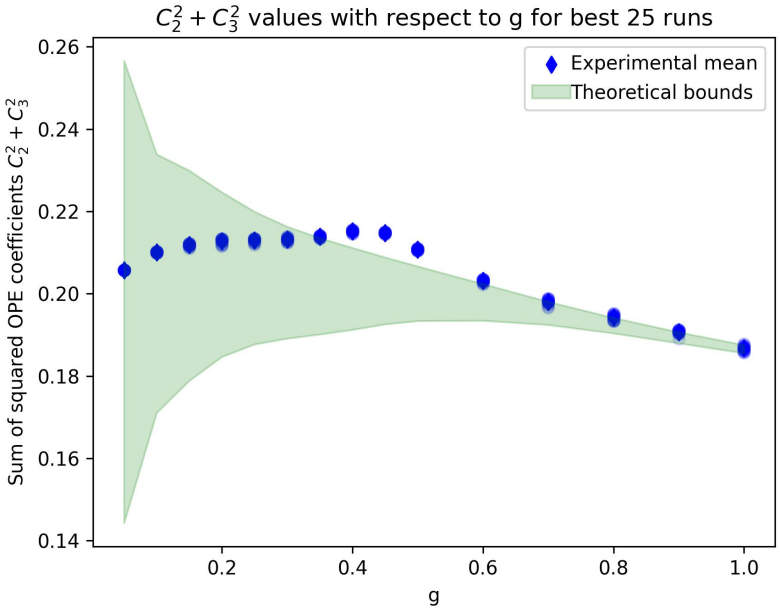


Figure 5.10: Theoretical bounds [CGJP22a] and experimental values for $C_2^2 + C_3^2$ in the one dimensional CFT model with $g \leq 1$. Only the top 25 runs are considered.

A possible explanation lies in the fact that as $g \to 0$ the scaling dimensions Δ_2, Δ_3 converge to the same value. Since the conformal blocks are analytical functions of Δ, their contribution to the conformal bootstrap equations are almost the same and can guide the agent in the wrong direction. A more in-depth discussion of this effect can be found in section 5.3.3 and in appendix A.2.1.

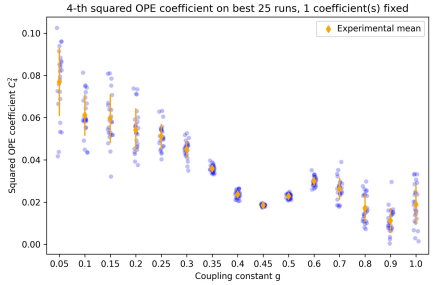
In the figures 5.8, 5.9 and 5.10 we also included the data for $g = 1$ as a reference for the behavior at strong coupling. In the latter case, the average values for C_2^2, C_3^2 are not inside the bound but the confidence intervals of the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$ both include the available region, meaning that for the strong coupling case we have more confidence on the results we obtain.

For completeness, in figure 5.11 we plot the other squared OPE coefficients C_i^2 with $4 \leq i \leq 10$ as functions of the coupling constant g.

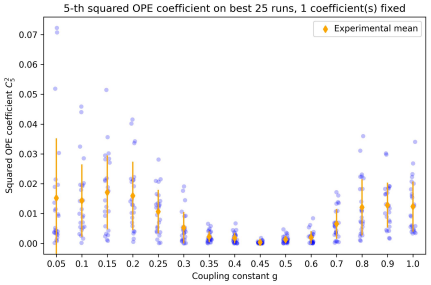
- In general, results show an increasing precision as the coupling constant grows as expected. This can be clearly seen for C_4^2 for which we expect the best results. This also confirms the fact that the search is most complex when $g \to 0$, confirming the higher uncertainty for these values of the coupling constant from [CGJP22a].

- Squared OPE coefficients C_9^2 and C_{10}^2 follow a very strange behavior, although they are related to the highest dimensional operators where the OPE is truncated. Therefore, we do not expect very good precision and relevance for these particular results.

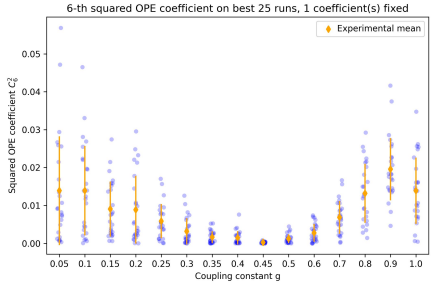
Figures A.4 and A.5 in appendix A.2 contain the same results, this time grouped by the value of g.



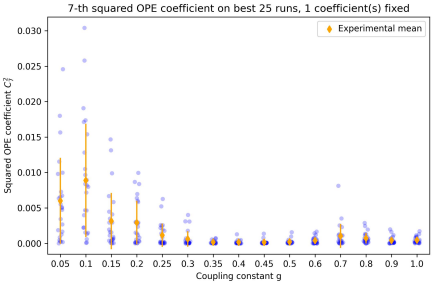
(a) C_4^2



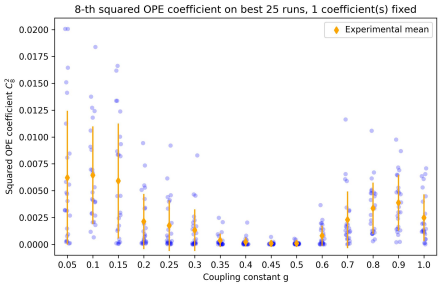
(b) C_5^2



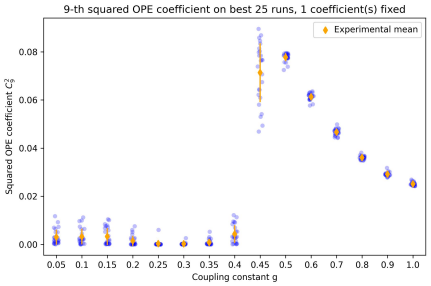
(c) C_6^2



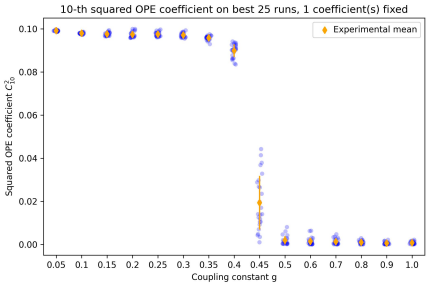
(d) C_7^2



(e) C_8^2



(f) C_9^2



(g) C_{10}^2

Figure 5.11: Experimental results on the unknown squared OPE coefficients C_i^2 as a function of the coupling constant g, one dimensional CFT model with $g \leq 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.

5.3.3 Experiments at strong coupling

In this section, we present the original results we were able to obtain for the one-dimensional CFT studied. In particular, we will study the squared OPE coefficients, indicated by C_i^2 with $i = 1, \dots, 10$ for values of the coupling $g \geq 1$. In this setting, the bounds on C_1^2, C_2^2, C_3^2 provided by [CGJP22b] are tight.

For the first squared OPE coefficient C_1^2, the same remarks as in section 5.3.2 apply, with the precision being sufficient to fix this value as input. For what concerns C_2^2 and C_3^2, the difference between the upper and the lower bounds is different between the strong coupling case and the weak coupling one. As it can be seen from figure 4.4, the bounds start very wide and end up being very small for coupling constants $g \geq 1$.

Further calculations show that, for $g = \frac{1}{2}$ and both C_2^2 and C_3^2, the distance between the middle point of the bounds and any other point in the available window is, at its lowest, around 3% of the lower bound. For weak coupling the window is wider for all values of $g \leq \frac{1}{2}$, even with both integral constraints applied, as we can clearly see from figure 4.4. The case of $\frac{1}{2} \leq g < 1$ has more precision, but not enough to make a confident choice on the actual value. Furthermore, as from figures 5.8 and 5.9, experimental results on C_2^2 and C_3^2 are not coherent with the theoretical bounds until $g = 1$, with more accuracy as the coupling constant grows.

On the other hand, at $g \geq 1$ the relative error on the second and third squared OPE coefficients go below 1% and the precision improves monotonically for $1 \leq g \leq 4$. To have confidence in our results, we therefore decided that for all coupling constants $g \geq 1$ the first three squared OPE coefficients would also be used as input for the algorithm, with $C_4^2, \dots, C_{10}^2$ being the only unknown CFT data for the search. This is in line with what we have discussed in section 4.4 and we will use the middle points of the bounds for C_1^2, C_2^2, C_3^2 as input values for the algorithm.

We ran experiments with the following coupling constants:

$$g = \{1, 1.5, 2, 2.5, 3, 3.5, 4\}. \tag{5.4}$$

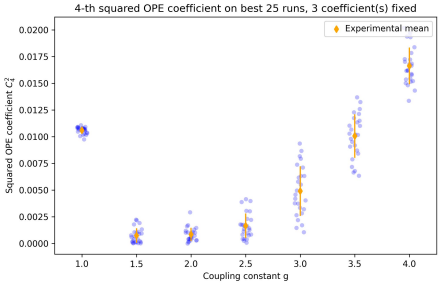
For each value, a total of 500 parallel runs were performed. Figure 5.12 shows the experimental results obtained in the 25 runs with the best overall rewards, with each squared OPE coefficient C_i^2 as a function of the coupling constant g. Light blue points represent the individual values for each experiment (one for each of the top 25 runs) whereas the yellow points represent the averages, with error bands of a single standard deviation.

As we can see, the results vary significantly between different values of g. Not only for the mere values found, which is expected, but mainly in terms of relative error. In particular, it is evident that the ratio $\frac{\text{std}(C_i^2)}{\text{avg}(C_i^2)}$ changes significantly between the different squared OPE coefficients C_i^2 considered. This can be seen as well in figure 5.13, where all C_i^2 for a fixed coupling constant are plotted.

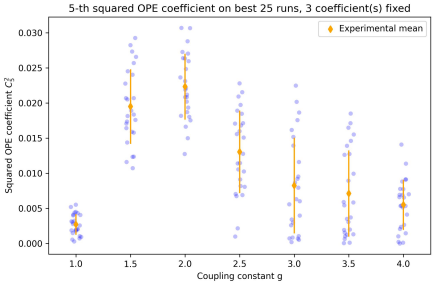
We refer to the appendix, section A.3, for the numerical values of the coefficients found. In particular, tables A.2, A.3, A.4 contain the results in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$.

We are now ready to comment on the results obtained:

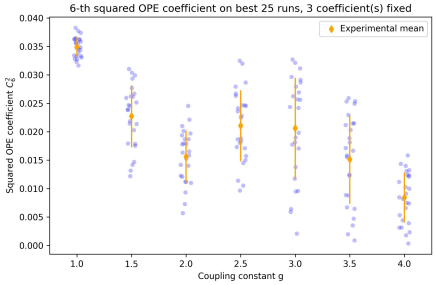
- The most precise results are the ones for the squared OPE coefficient C_4^2, which corresponds to the lowest dimensional operator with unknown data. This is indeed something to expect since low-dimensional operators have the highest influence on the crossing equation.



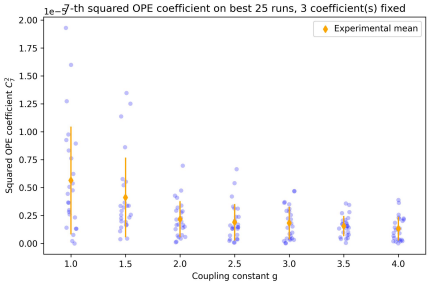
(a) C_4^2



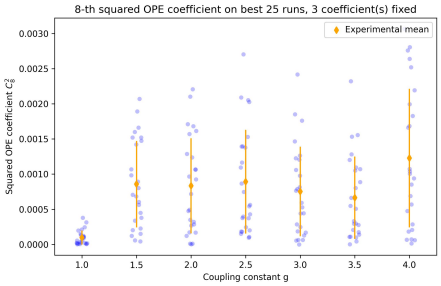
(b) C_5^2



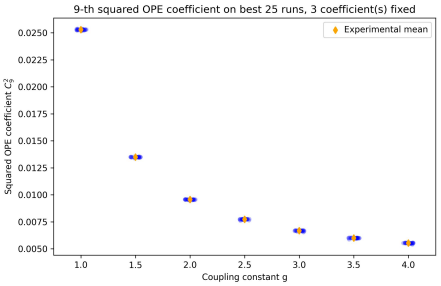
(c) C_6^2



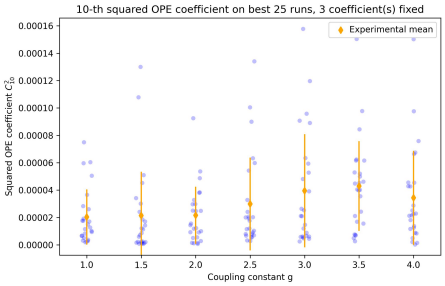
(d) C_7^2



(e) C_8^2

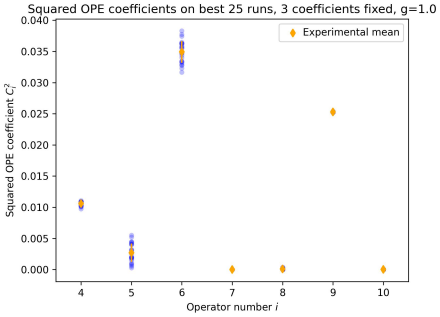


(f) C_9^2

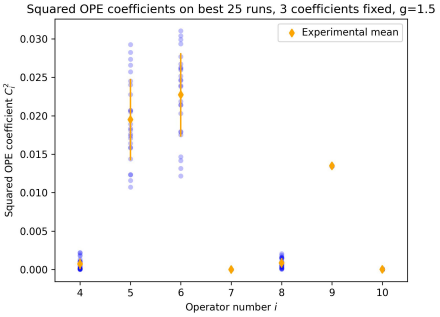


(g) C_{10}^2

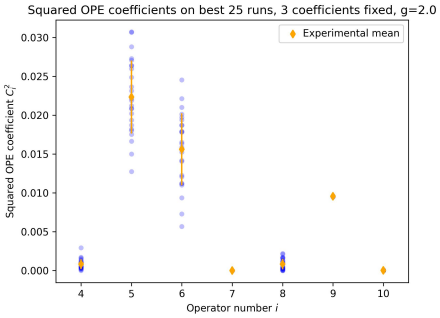
Figure 5.12: Experimental results on the unknown squared OPE coefficients C_i^2 as a function of the coupling constant g, one dimensional CFT model with $g \geq 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.



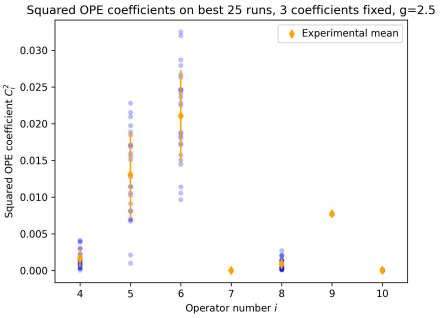
(a) $g = 1$



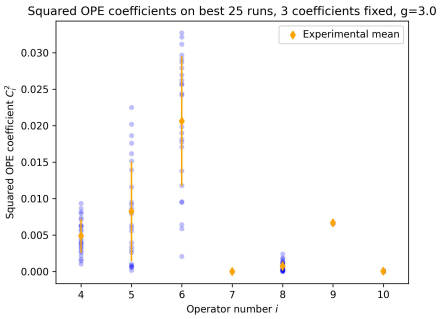
(b) $g = 1.5$



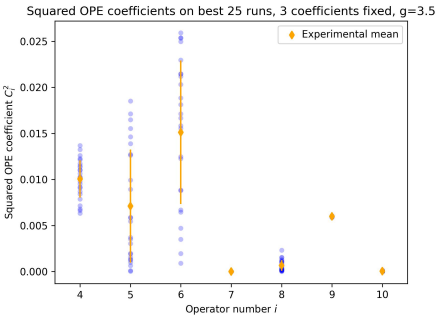
(c) $g = 2$



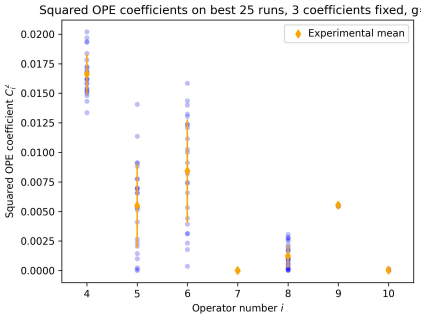
(d) $g = 2.5$



(e) $g = 3$



(f) $g = 3.5$



(g) $g = 4$

Figure 5.13: Experimental results on the unknown squared OPE coefficients C_i^2, one dimensional CFT model with $g \geq 1$. Results are grouped by the value of the coupling constant g. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.

- Results for C_5^2 and C_6^2 start well at $g = 1$ but lose precision as g increases. This is counter-intuitive, since previous results on C_1^2, C_2^2, C_3^2 with other methodologies [CGJP22a] show the opposite behavior. We shortly try to find an explanation for this.

- The squared OPE coefficient C_7^2 shows instead an increased precision as g grows, although this could also be related to the fact that values seem to converge to 0. Indeed, previous research with perturbative approaches [FM21] showed that, up to the 4-th perturbative order, $C_7^2 \to 0$ as $g \to \infty$. Our experimental results seem to confirm this fact.

- Results for C_8^2 and C_{10}^2 are the least accurate overall in terms of relative error, while C_9^2 has surprisingly high precision. We should also keep in mind that results for these operators, the ones with the highest scaling dimension, are the ones with the least expected precision.

In general, squared OPE coefficients seem to follow an interesting pattern: the relative error increases with g and we seek to find a possible explanation for this. Let us now recall the expression of the conformal block decomposition in the model:

$$f(x) = F_{\mathcal{I}}(x) + C_{BPS}^2 F_{\mathcal{B}_2}(x) + \sum_{n=1}^{\infty} C_n^2 F_{\Delta_n}(x). \tag{5.5}$$

Focusing on the last sum, which contains the conformal blocks with the unknown CFT data, recall that $F_{\Delta_n}(x)$ is

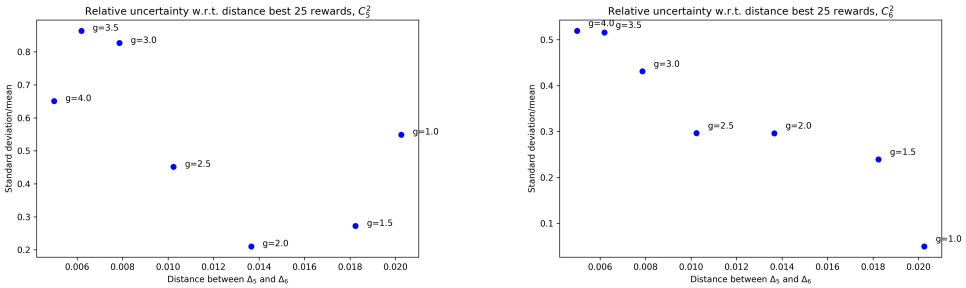
$$F_{\Delta}(x) = \frac{x^{\Delta+1}}{1-\Delta}\, {}_2F_1(\Delta+1, \Delta+2, 2\Delta+4; x) \tag{5.6}$$

which is an analytical function of the scaling dimension Δ.

Taking a look back at figure 4.1, we can see that many of the scaling dimensions converge to the same values as $g \to \infty$. In particular, Δ_5 and Δ_6 are almost everywhere overlapping in the graph, with their distance becoming even smaller as g increases. This leads to the fact that F_{Δ_5} and F_{Δ_6} have very close values for every x and, hence, could be indistinguishable by the algorithm. With an abuse of notation, we can rewrite this observation as

$$C_5^2 F_{\Delta_5} + C_6^2 F_{\Delta_6} \approx (C_5^2 + C_6^2) F_{\Delta_5}, \tag{5.7}$$

with the approximation becoming more and more trustworthy as g grows. Hence, the squared OPE coefficients C_5^2 and C_6^2 are in some sense correlated and, if we consider the following transformation $C_5^2 \mapsto C_5^2 + \delta, C_6^2 \mapsto C_6^2 - \delta$ with $\delta \in \mathbb{R}$ small, formula (5.5), remains almost unchanged within this approximation. It seems therefore reasonable that the uncertainty on Δ_5 and Δ_6 increases as the distance between Δ_5 and Δ_6 decreases. This can be seen in figure 5.14 where we plotted the ratios $\frac{\text{std}(C_5^2)}{\text{avg}(C_5^2)}$ and $\frac{\text{std}(C_6^2)}{\text{avg}(C_6^2)}$ as functions of $|\Delta_5 - \Delta_6|$.



(a) Normalized uncertainty on C_5^2 (b) Normalized uncertainty on C_6^2

Figure 5.14: Relative error on C_5^2 (left) and C_6^2 (right) as a function of the distance between Δ_5 and Δ_6 for the one dimensional model, $g \geq 1$.

Furthermore, being

$$C_5^2 F_{\Delta_5} + C_6^2 F_{\Delta_6} \approx (C_5^2 + C_6^2) F_{\Delta_5} \tag{5.8}$$

we should also have that $(C_5^2 + C_6^2)$ has very good precision since it is the coefficient of this combined conformal block. In fact, image 5.15 shows that the uncertainty on $C_5^2 + C_6^2$ is lower and comparable for almost every g.

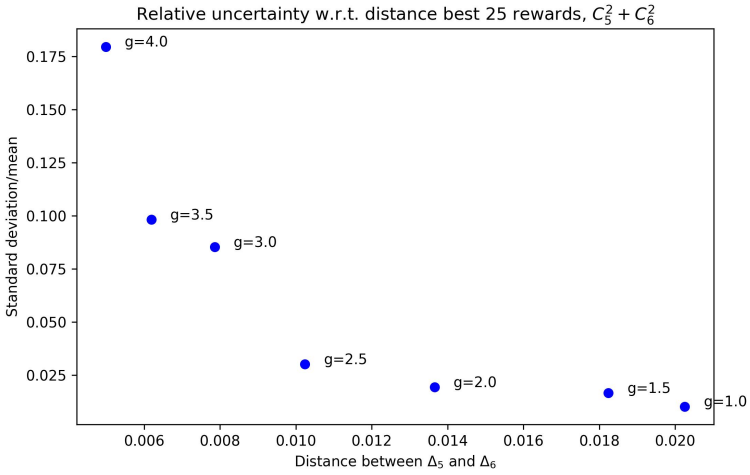
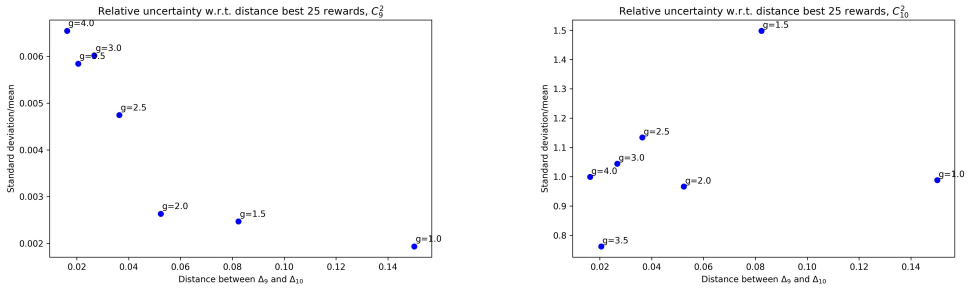


Figure 5.15: Relative error on $C_5^2 + C_6^2$ as a function of the distance between Δ_5 and Δ_6 for the one dimensional model, $g \geq 1$

Similar reasoning is also valid for C_9^2 and C_{10}^2, as it can be seen from images 5.16 and 5.17. We are truncating the crossing equation at the tenth operator and we do not expect much precision on Δ_9 and Δ_{10}. For some reason, the algorithm gives an almost perfect precision on C_9^2 and complete uncertainty on C_{10}^2, although there is currently no explanation for this behavior.



(a) Normalized uncertainty on C_9^2 (b) Normalized uncertainty on C_{10}^2

Figure 5.16: Relative error on C_9^2 (left) and C_{10}^2 (right) as a function of the distance between Δ_9 and Δ_{10} for the one dimensional model, $g \geq 1$.

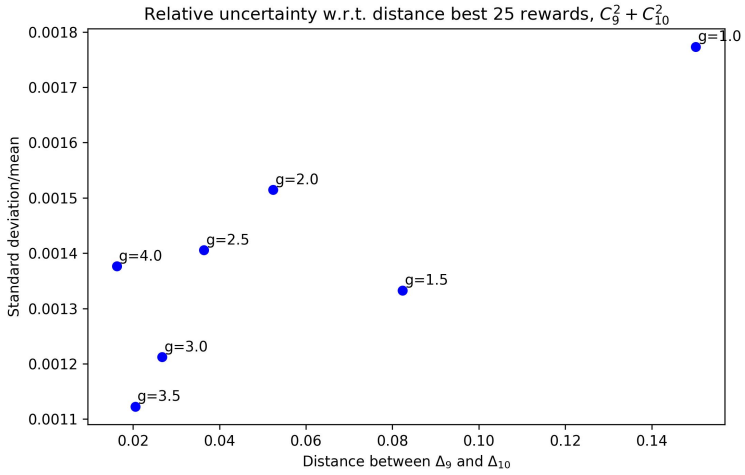


Figure 5.17: Relative error on $C_9^2 + C_{10}^2$ as a function of the distance between Δ_9 and Δ_{10} for the one dimensional model, $g \geq 1$

Finally, we refer to the appendix, section A.2, for a similar analysis on C_2^2, C_3^2 for the weak coupling case.

5.4 Discussion

The Bootstrap Stochastic Optimization algorithm is a useful tool for finding the unknown CFT data in the crossing equation, although it needs some precautions. The nature of the Reinforcement Learning algorithm requires us to consider as most promising results the ones with the highest reward obtained by the agent. In fact, when no analytical or numerical values are available, the reward is the only metric we have to judge the results since it expresses how the crossing equation is closer to zero with the estimates of the CFT data found.

For the Ising $2D$ model:

- **The free search is a hard task and can be misleading**. The highest reward obtained was associated to values of $(\vec{\Delta}, \vec{C^2})$ very different from the theoretical

ones which, plugged directly into the formula, provided a smaller value for the reward.

- **High errors may be related to the approximation we have made**. The original crossing equation is truncated up to a Δ_{max} and is tested only on a set of $N_z = 180$ points in the complex plane, not on the whole range of admissible cross-ratio values.

- **Constraining the values of the scaling dimensions improves the results**. The smaller search space, related to a simplified landscape of the crossing equation helps the algorithm to find the correct CFT data.

- **Constraining the problem increases the stability and the precision as well**. Results are more stable and the precision increases as we increase the number of scaling dimensions given as input. Since in the one-dimensional CFT the scaling dimensions for the first 10 operators are known, the results can be promising.

We then moved to the actual model of interest: the one-dimensional CFT. In all the experiments we gave as input the scaling dimension values Δ_i for all the operators.

With some reward engineering, we determined that the best form for the reward was given by the formula R_2 in (4.38), in line with our expectation. This formula pushes all three constraints, the crossing equation and the two integral ones, to move toward zero in order to increase the reward. Integral constraints seem easier to optimize, as the optimal weights found are $w_1 = 10^4, w_2 = 10^5$ and, in the experiments using the reward R_1 in 4.38, the components in the sum corresponding to the integral constraints were the first to grow. In general, it seems that this approach of optimizing them at the same time at the denominator is more efficient.

As seen from figure 5.7, the agent also successfully learns to achieve greater rewards as the exploration goes on, which also means that it can find good estimates for the solution of the crossing equation and the integral constraints. Results are also very stable, meaning that we can trust the estimates obtained without performing a huge number of parallel runs.

To increase even more the chance of finding meaningful results we added as input the first squared OPE coefficient for all values of the coupling constant g for the $1D$ CFT model.

For the one-dimensional model:

- **Values of the coupling constant $g < 1$ are the hardest to explore**. Our estimates for C_2^2 and C_3^2 are always outside the expected range for $g < 1$. In section A.2 we also try to develop further including an analysis on the sum $C_2^2 + C_3^2$ but the weak coupling case remains still hard to tackle. Although very precise, results are not accurate or promising compared to other approaches.

- **At strong coupling, we were able to produce interesting results**. As expected, the results with the highest precision are related to the lowest dimensional operators, with C_4^2 being clearly the most accurate. Similarly, except for C_9^2 which makes a strange and unmotivated exception, higher dimensional operators show the least precision. This was expected, as these are also the operators where the truncation of the OPE is performed and they could be affected by this approximation.

- C_7^2 **seem to converge to** 0 **as** g **increases**. From figure 5.12 it seems that $C_7^2 \to 0$ as $g \to \infty$. This fact confirms the results in [FM21], where it was proven up to the 4-th perturbative degree.

- **Similar scaling dimensions are related to lower precision**. We also investigated the results on the precision for operators with very similar scaling dimensions. When the distance between Δ_i and Δ_j is small, the algorithm tends to produce estimates of C_i^2 and C_j^2 with low precision, with their sum being more accurate. We showed as an example the cases of C_5^2, C_6^2 and C_9^2, C_{10}^2 and we also tried to explain this behavior in section 5.3.3.

Chapter 6

Conclusion

In this thesis, we studied how Reinforcement Learning can be applied to solve a physical equation related to the Conformal Field Theory. In particular, we developed a particular implementation of the Soft Actor-Critic algorithm and translated the CFT setting into an RL environment. The conformal bootstrap equation, the main equation we want to solve, is evaluated on a set of complex points from which we calculate the reward and the information needed for the agent to learn how to solve this optimization problem.

In chapter 4 we described the Bootstrap Stochastic Optimization algorithm, as well as all the remarks required to implement and use the algorithm in the CFT framework. The two models we applied the algorithms on are the $2D$ Ising model, for which an analytical solution for the crossing equation is available, and the $1D$ defect CFT defined on the $\frac{1}{2}$ Wilson line in the $4D$ $N = 4$ super Yang-Mills theory.

While the Ising $2D$ model is used as a toy model to validate the approach, this work is the first application of RL to the one-dimensional model studied. In the $1D$ defect CFT, previous results were able to find with sufficient numerical precision the values for the first 10 scaling dimensions. Theoretical bounds on C_1^2, C_2^2, C_3^2 are also available, but the precision depends on the main parameter of the model: the coupling constant g. This theory has also two additional integral constraints on the 4-point function that can be enforced. With some reward engineering, we found a way to integrate these additional constraints into the framework to constrain the search even more.

For the $2D$ Ising model we showed that if no additional information is provided and the search is performed on all 22 (or more) unknowns of the crossing equation, the results can be misleading, as the CFT data obtained by the agent corresponds to a different solution from the analytical one but with a greater reward. We then constrained the search by giving as input to the algorithm an increasing number of scaling dimensions and the results started improving. The CFT data we obtained was more accurate and precise with increased stability in the rewards as well. This setting replicates more accurately our original experiments on the one-dimensional model, where all the values of Δ are given as input. With these improvements, we are more confident that the results for the next experiments can be promising, even if there is no complete analytical solution available.

We finally studied $1D$ defect CFT, making a distinction between the weak coupling and the strong coupling cases. For small values of g, with the additional input of the first squared OPE coefficient C_1^2, we showed that our estimates were outside the expected range for C_2^2, C_3^2, although the relative error was small for all $g < 1$. The

small coupling case remains hard to tackle.

For the strong coupling case, on the other hand, the theoretical bounds for C_2^2, C_3^2 are tight enough to give their average values as input to the algorithm, leading to a simpler optimization problem. We studied the squared OPE coefficients and their precision as a function of the coupling constant with $g \geq 1$ and we were able to obtain estimates for $C_i^2, 4 \leq i \leq 10$, for which no previous accurate results were available. As expected, the highest dimensional operators near the cutoff are related to the least precision, while the most accurate results were for C_4^2.

We also noticed that couples of operators with very similar scaling dimensions were related to a lower precision, while the sum between their squared OPE coefficient had a lower relative error. We tried to give an explanation for this observation by leveraging the fact that the conformal blocks are analytical functions.

Finally, we also noted that C_7^2 seems to converge to 0 as $g \to \infty$, in agreement with previous results with perturbative approaches.

Possible future works in this direction include the investigation of higher dimensional operators and the integration of multiple crossing equations (related to different operators) into the same framework. Additional constraints we include in the algorithm can significantly improve the precision as previous results have shown. From an RL and ML point of view, we can improve the performance and make the search easier by feeding the agent with additional information on the operator he is currently acting on. As of now, operators are treated equally, with the agent identifying them with only the repetitive cycling between them. Finally, physically informed models and neural networks can enforce physical constraints or learn physical laws automatically and are a concrete possibility to improve the results.

Appendix A

Further results

A.1 $\langle\sigma\sigma\sigma\sigma\rangle$ correlator in the Ising $2D$ model

The $\langle\sigma\sigma\sigma\sigma\rangle$ correlator leads to a more complex problem than the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator: it involves a total of 16 conformal blocks with $\Delta \leq \Delta_{\max} = 10.5$, leading to a total of 32 unknowns. The results found are worse than the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator, as we would expect. The reward obtained from the theoretical values is 10590097.015 while the average for the top 25 runs and the best individual run are respectively 3694.7704 and 5575.033, various orders of magnitudes smaller. In figures A.1 and A.2 it seems also that the agent had not the possibility to explore widely the solution space, especially for the scaling dimensions. The Δ values are seemingly close to each other for the top 25 runs while also being very far away from the analytical solution. This could be explained by the use of non-optimal parameters since we did not perform a grid search for this correlator in particular. This was not the main focus of this work and, being the experiment time demanding, we chose not to study this correlator further.

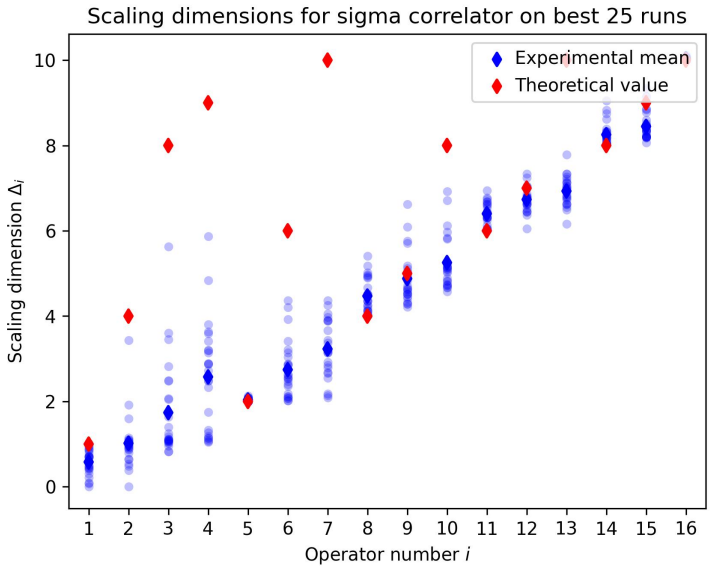


Figure A.1: Results for the scaling dimensions Δ_i of the operators in the $\langle\sigma\sigma\sigma\sigma\rangle$ correlator, $2D$ Ising model. Red points represent the theoretical values and blue points represent experimental values for the 25 runs with the best reward.

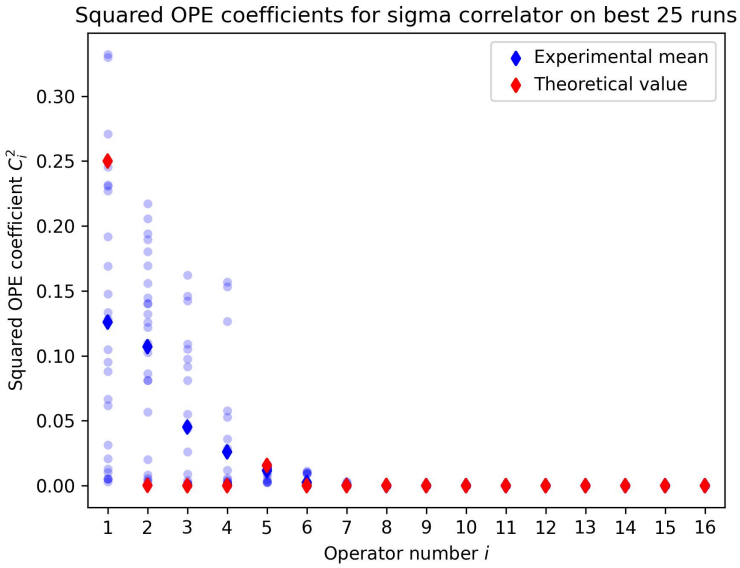


Figure A.2: Results for the squared OPE coefficients C_i^2 of the operators in the $\langle\sigma\sigma\sigma\sigma\rangle$ correlator, $2D$ Ising model. Red points represent the theoretical value and blue points represent experimental values for the 25 runs with the best reward.

For what concerns the constrained search experiments, results were worse than the ones for the $\langle\epsilon\epsilon\epsilon\epsilon\rangle$ correlator as expected. However, even for this correlator, the relative error on the squared OPE coefficients decreases as we give more Δ's as input as we can see from figure A.3.

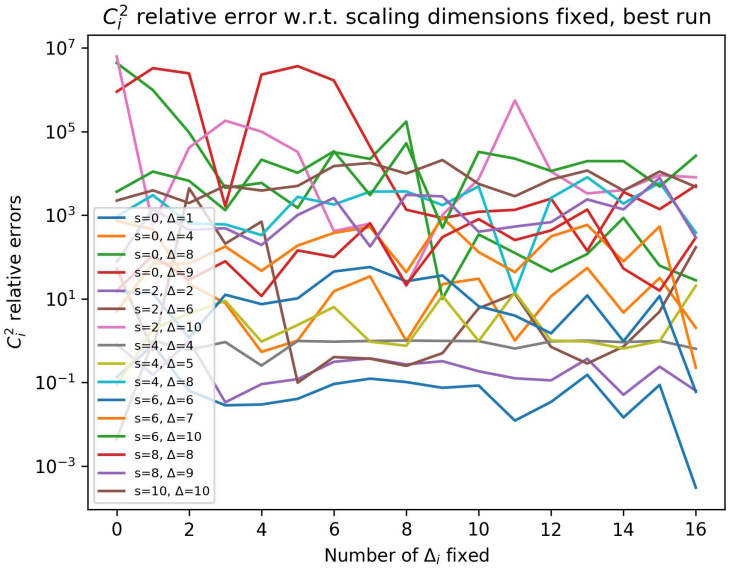


Figure A.3: Relative error on the squared OPE coefficients C_i^2 as a function of the number of scaling dimensions Δ_i given as input. $\langle\sigma\sigma\sigma\sigma\rangle$ correlator in the $2D$ Ising model. Only the best run for each case is considered.

A.2 Experiments at weak coupling

In this section, we provide the numerical values and some additional graphs for the results at weak coupling in section 5.3.2. In table A.1 we collect the numerical estimates of C_2^2, C_3^2 obtained in our experiments, together with the corresponding bounds from [CGJP22b]. Our results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$, where the mean and the standard deviation (or uncertainty) are calculated on the 25 runs that obtained the best reward.

	C_2^2 **Lower**	C_2^2 **Upper**	C_2^2 **experimental**	C_3^2 **Lower**	C_3^2 **Upper**	C_3^2 **experimental**
$g = 0.05$	0.078952	0.133857	$8.611 \pm 15.708 \times 10^{-5}$	0.065284	0.122818	$2.057 \pm 0.001 \times 10^{-2}$
$g = 0.10$	0.085494	0.114075	$5.993 \pm 0.063 \times 10^{-2}$	0.085575	0.119813	$1.501 \pm 0.007 \times 10^{-1}$
$g = 0.15$	0.086939	0.107832	$7.898 \pm 0.062 \times 10^{-2}$	0.091939	0.122087	$1.328 \pm 0.009 \times 10^{-1}$
$g = 0.20$	0.087271	0.101766	$8.225 \pm 0.062 \times 10^{-2}$	0.097415	0.122897	$1.305 \pm 0.012 \times 10^{-1}$
$g = 0.25$	0.084145	0.094569	$7.924 \pm 0.035 \times 10^{-2}$	0.103561	0.125371	$1.337 \pm 0.008 \times 10^{-1}$
$g = 0.30$	0.078634	0.086697	$7.297 \pm 0.028 \times 10^{-2}$	0.110483	0.129605	$1.402 \pm 0.007 \times 10^{-1}$
$g = 0.35$	0.072357	0.078986	$6.57 \pm 0.016 \times 10^{-2}$	0.117757	0.134551	$1.485 \pm 0.004 \times 10^{-1}$
$g = 0.40$	0.066450	0.072044	$5.736 \pm 0.019 \times 10^{-2}$	0.124799	0.139115	$1.578 \pm 0.005 \times 10^{-1}$
$g = 0.45$	0.061380	0.066057	$5.077 \pm 0.015 \times 10^{-2}$	0.131208	0.142791	$1.640 \pm 0.003 \times 10^{-1}$
$g = 0.50$	0.057203	0.061181	$4.765 \pm 0.015 \times 10^{-2}$	0.136181	0.148447	$1.631 \pm 0.003 \times 10^{-1}$
$g = 0.60$	0.050998	0.053931	$4.437 \pm 0.038 \times 10^{-2}$	0.142468	0.148446	$1.587 \pm 0.007 \times 10^{-1}$
$g = 0.70$	0.046696	0.048706	$4.172 \pm 0.083 \times 10^{-2}$	0.145739	0.149352	$1.563 \pm 0.014 \times 10^{-1}$
$g = 0.80$	0.043568	0.045003	$3.901 \pm 0.11 \times 10^{-2}$	0.146815	0.149150	$1.552 \pm 0.017 \times 10^{-1}$
$g = 0.90$	0.041211	0.042279	$3.726 \pm 0.12 \times 10^{-2}$	0.146769	0.148377	$1.534 \pm 0.015 \times 10^{-1}$
$g = 1.00$	0.039378	0.040198	$3.819 \pm 0.16 \times 10^{-2}$	0.146175	0.147339	$1.485 \pm 0.022 \times 10^{-1}$

Table A.1: Theoretical bounds [CGJP22a] and experimental values for C_2^2 and C_3^2 in the one dimensional CFT model with $g \leq 1$. Only the top 25 runs are considered. Experimental results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$.

Figures A.4 and A.5 contain the the experimental results for the squared OPE coefficients C_i^2 for $4 \leq i \leq 10$ grouped by the coupling constant g.

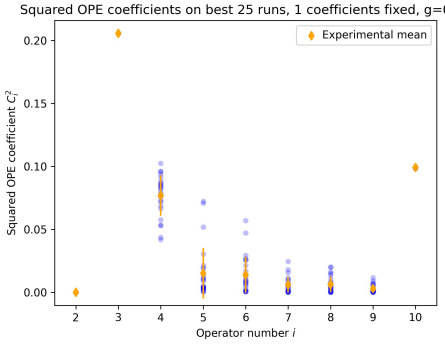
A.2.1 Second and third squared OPE coefficients

Recall that, in section 5.3.3, we tried to give an explanation for the loss in precision for C_i^2 values related to operators with very similar squared OPE coefficients. From figure 4.1 we can also see that, as $g \to 0$, the values for Δ_2 and Δ_3 tend towards the same value of 2. This may be one of the root causes of the loss in accuracy we found for C_2^2 and C_3^2. In fact, our experimental data show that the experimental values of these OPE coefficients are outside the expected bounds as in figures 5.8, 5.9.

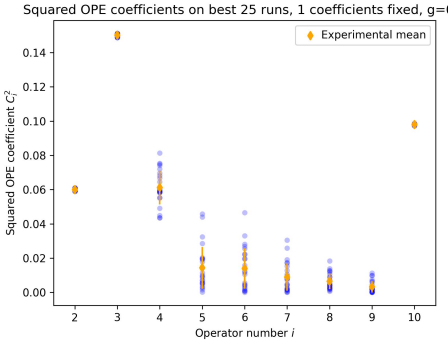
We then want to investigate if the sum of the squared OPE coefficients $C_2^2 + C_3^2$ is inside the sum of the theoretical bounds found in [CGJP22a]. Figure 5.10 represents the experimental data for $C_2^2 + C_3^2$, with the available region being determined by the set sum of the individual ones.

As we can see, most experimental values are now inside the bounds which are now very wide. The situation remains unclear since coupling constants close to $g \sim 0.4$ have a greater distance between Δ_2 and Δ_3, but the corresponding sums $C_2^2 + C_3^2$ lie outside the available regions.

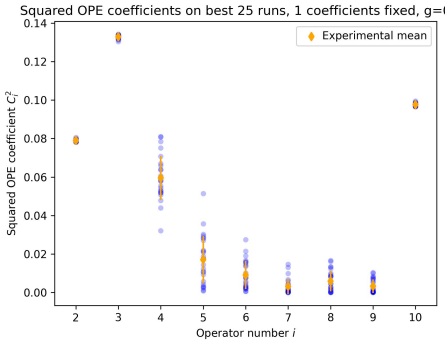
This confirms the difficulty in finding a good approach for the weak coupling case.



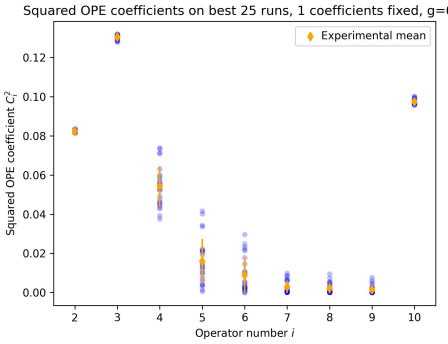
(a) $g = 0.05$



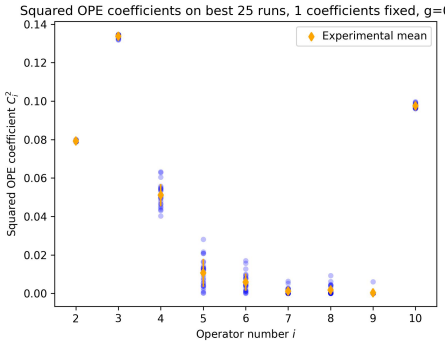
(b) $g = 0.10$



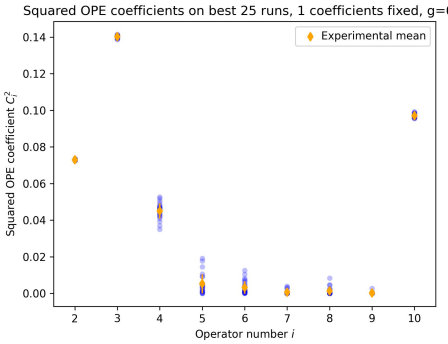
(c) $g = 0.15$



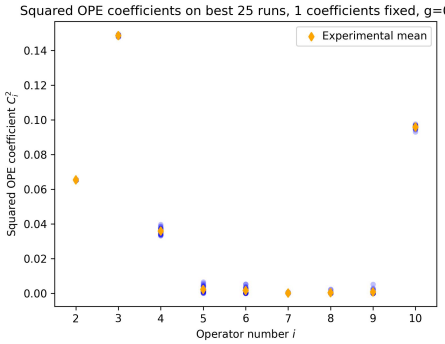
(d) $g = 0.20$



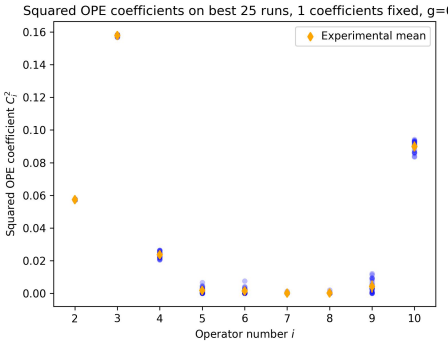
(e) $g = 0.25$



(f) $g = 0.30$

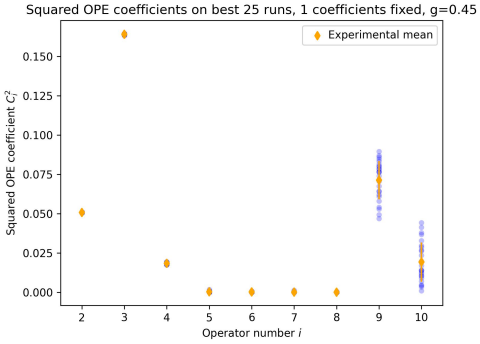


(g) $g = 0.35$

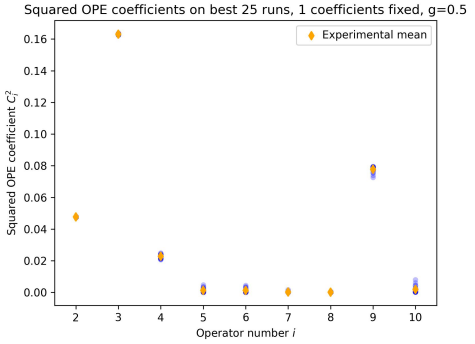


(h) $g = 0.40$

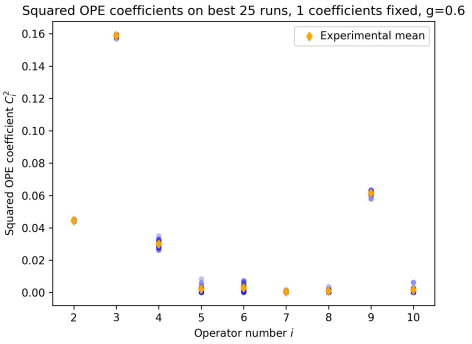
Figure A.4: Experimental results on the unknown squared OPE coefficients C_i^2 as a function of the coupling constant g, one dimensional CFT model with $g \leq 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.



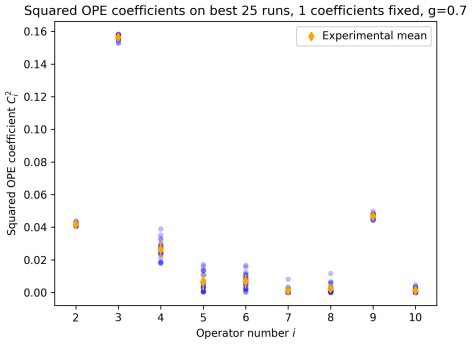
(a) $g = 0.45$



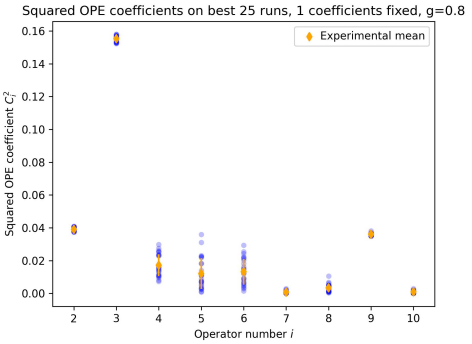
(b) $g = 0.50$



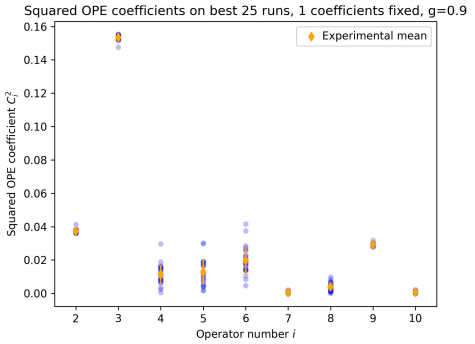
(c) $g = 0.60$



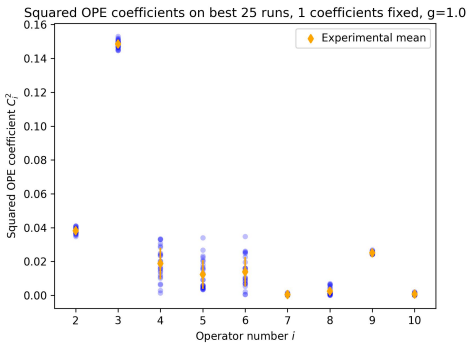
(d) $g = 0.70$



(e) $g = 0.80$



(f) $g = 0.90$



(g) $g = 1.00$

Figure A.5: Experimental results on the unknown squared OPE coefficients C_i^2 as a function of the coupling constant g, one dimensional CFT model with $g \leq 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.

A.3 Experiments at strong coupling

In tables A.2, A.3, A.4 we summarize the results found for the 25 runs that got the best reward for each g in the strong coupling case, as described in section 5.3.3. The results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$, where the average and standard deviation are calculated on the selected runs.

	$g = 1.0$	$g = 1.5$
C_4^2	$1.059 \pm 0.0313 \times 10^{-2}$	$7.319 \pm 6.862 \times 10^{-4}$
C_5^2	$2.696 \pm 1.480 \times 10^{-3}$	$1.950 \pm 0.531 \times 10^{-2}$
C_6^2	$3.493 \pm 0,174 \times 10^{-2}$	$2.273 \pm 0.544 \times 10^{-2}$
C_7^2	$5.640 \pm 4.825 \times 10^{-6}$	$4.109 \pm 3.564 \times 10^{-6}$
C_8^2	$0.966 \pm 1.011 \times 10^{-4}$	$8.573 \pm 6.146 \times 10^{-4}$
C_9^2	$2.529 \pm 0.0049 \times 10^{-2}$	$1.349 \pm 0.0033 \times 10^{-2}$
C_{10}^2	$2.035 \pm 2.011 \times 10^{-5}$	$2.141 \pm 3.207 \times 10^{-5}$

Table A.2: Experimental values for the unknown squared OPE coefficients in the one dimensional, CFT model, $g = 1, 1.5$. Only the top 25 runs are considered. Experimental results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$.

	$g = 2.0$	$g = 2.5$	$g = 3.0$
C_4^2	$8.346 \pm 6.310 \times 10^{-4}$	$1.648 \pm 1.155 \times 10^{-3}$	$4.896 \pm 2.322 \times 10^{-3}$
C_5^2	$2.233 \pm 0.470 \times 10^{-2}$	$1.303 \pm 0.589 \times 10^{-2}$	$8.239 \pm 6.812 \times 10^{-3}$
C_6^2	$1.561 \pm 0.462 \times 10^{-2}$	$2.105 \pm 0.624 \times 10^{-2}$	$2.061 \pm 0.889 \times 10^{-2}$
C_7^2	$2.171 \pm 1.611 \times 10^{-6}$	$1.894 \pm 1.631 \times 10^{-6}$	$1.823 \pm 1.460 \times 10^{-6}$
C_8^2	$8.335 \pm 6.771 \times 10^{-4}$	$8.915 \pm 7.371 \times 10^{-4}$	$7.519 \pm 6.371 \times 10^{-4}$
C_9^2	$9.557 \pm 0.0252 \times 10^{-3}$	$7.723 \pm 0.0366 \times 10^{-3}$	$6.670 \pm 0.0401 \times 10^{-3}$
C_{10}^2	$2.162 \pm 2.089 \times 10^{-5}$	$2.991 \pm 3.391 \times 10^{-5}$	$3.953 \pm 4.130 \times 10^{-5}$

Table A.3: Experimental values for the unknown squared OPE coefficients in the one dimensional, CFT model, $g = 2, 2.5, 3$. Only the top 25 runs are considered. Experimental results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$.

	$g = 3.5$	$g = 4.0$
C_4^2	$1.006 \pm 0.204 \times 10^{-2}$	$1.662 \pm 0.171 \times 10^{-2}$
C_5^2	$7.107 \pm 6.139 \times 10^{-3}$	$5.491 \pm 3.573 \times 10^{-3}$
C_6^2	$1.512 \pm 0.779 \times 10^{-3}$	$8.419 \pm 4.374 \times 10^{-3}$
C_7^2	$1.550 \pm 0.916 \times 10^{-6}$	$1.323 \pm 1.070 \times 10^{-6}$
C_8^2	$6.638 \pm 5.866 \times 10^{-4}$	$1.228 \pm 0.987 \times 10^{-3}$
C_9^2	$5.992 \pm 0.0350 \times 10^{-3}$	$5.526 \pm 0.0362 \times 10^{-3}$
C_{10}^2	$4.296 \pm 3.275 \times 10^{-5}$	$3.439 \pm 3.437 \times 10^{-5}$

Table A.4: Experimental values for the unknown squared OPE coefficients in the one dimensional, CFT model, $g = 3.5, 4$. Only the top 25 runs are considered. Experimental results are in the form $\text{avg}(C_i^2) \pm \text{std}(C_i^2)$.

A.4 Higher dimensional operators in the 1D CFT

We performed an additional experiment including as unknowns the CFT data of higher dimensional operators. In fact, we added 5 terms to the expansion of the conformal blocks (4.24), obtaining the following

$$f(x) = F_{\mathcal{I}}(x) + C^2_{BPS} F_{\mathcal{B}_2}(x) + \sum_{n=1}^{15} C^2_n F_{\Delta_n}(x) \tag{A.1}$$

We only considered the case of a coupling constant $g = 1$ and we fixed both the first 10 scaling dimensions $\Delta_n, 1 \leq n \leq 10$ and C^2_1, C^2_2, C^2_3 as already discussed in order to reduce the complexity of this experiment as much as possible.

Unfortunately, results on the unknown operators from the 11-th to the 15-th are not precise at all as can be seen from figures A.6 and A.7.

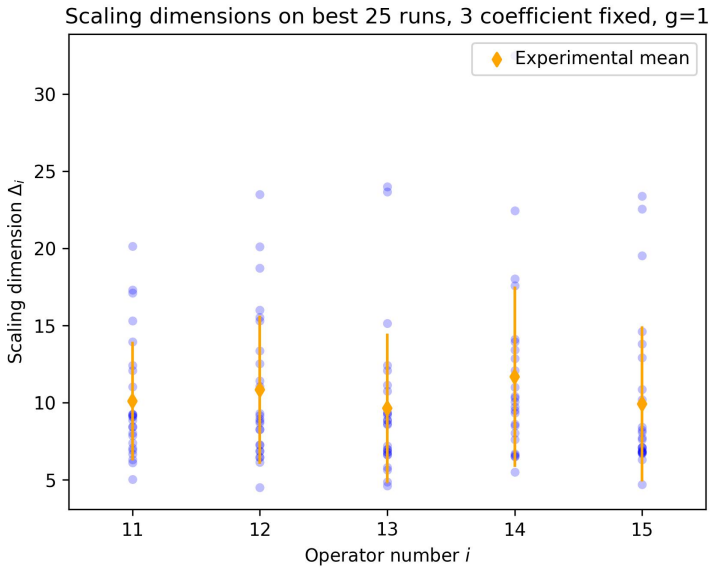


Figure A.6: Experimental results on the unknown scaling dimensions Δ_i of higher dimensional operators in the one-dimensional CFT model with $g = 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.

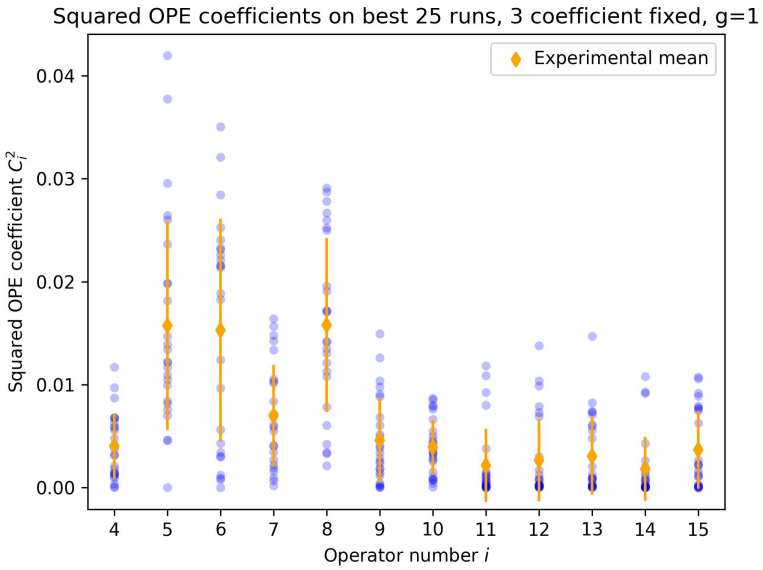


Figure A.7: Experimental results on the unknown squared OPE coefficients C_i^2 of higher dimensional operators in the one-dimensional CFT model with $g = 1$. Only the best 25 runs are considered. Blue points represent the individual value for each run, while yellow points and bars represent their means and standard deviations.

Bibliography

[BAA+11] Niklas Beisert, Changrim Ahn, Luis F. Alday, Zoltàn Bajnok, James M. Drummond, Lisa Freyhult, Nikolay Gromov, Romuald A. Janik, Vladimir Kazakov, Thomas Klose, Gregory P. Korchemsky, Charlotte Kristjansen, Marc Magro, Tristan McLoughlin, Joseph A. Minahan, Rafael I. Nepomechie, Adam Rej, Radu Roiban, Sakura Schäfer-Nameki, Christoph Sieg, Matthias Staudacher, Alessandro Torrielli, Arkady A. Tseytlin, Pedro Vieira, Dmytro Volin, and Konstantinos Zoubos. Review of AdS/CFT integrability: An overview. *Letters in Mathematical Physics*, 99(1-3):3–32, oct 2011.

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[Bot91] Françoise (Advisor (for a thesis or dissertation)) Bottou, Léon; Fogelman. *Une approche théorique de l'apprentissage connexionniste et applications à la reconnaissance de la parole.* PhD thesis, Université de Paris 11, Orsay, France Degree-grantor, 1991.

[BPZ84] A.A. Belavin, A.M. Polyakov, and A.B. Zamolodchikov. Infinite conformal symmetry in two-dimensional quantum field theory. *Nuclear Physics B*, 241(2):333–380, 1984.

[CGJP22a] Andrea Cavaglià, Nikolay Gromov, Julius Julius, and Michelangelo Preti. Bootstrability in defect CFT: integrated correlators and sharper bounds. *Journal of High Energy Physics*, 2022(5), may 2022.

[CGJP22b] Andrea Cavaglià, Nikolay Gromov, Julius Julius, and Michelangelo Preti. Integrability and conformal bootstrap: One dimensional defect conformal field theory. *Physical Review D*, 105(2), jan 2022.

[CT12] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley, 2012.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[DK06a] Nadav Drukker and Shoichi Kawamoto. Small deformations of supersymmetric wilson loops and open spin-chains. *Journal of High Energy Physics*, 2006(07):024–024, jul 2006.

[DK06b] Nadav Drukker and Shoichi Kawamoto. Small deformations of supersymmetric wilson loops and open spin-chains. *Journal of High Energy Physics*, 2006(07):024–024, jul 2006.

[DKN+19] Patrick Dorey, Gregory Korchemsky, Nikita Nekrasov, Volker Schomerus, Didina Serban, and Leticia Cugliandolo. *Integrability: From Statistical Systems to Gauge Theory: Lecture Notes of the Les Houches Summer School: Volume 106, June 2016*. Oxford University Press, 07 2019.

[DO01] F.A. Dolan and H. Osborn. Conformal four point functions and the operator product expansion. *Nuclear Physics B*, 599(1-2):459–496, apr 2001.

[DO04] F.A. Dolan and H. Osborn. Conformal partial waves and the operator product expansion. *Nuclear Physics B*, 678(1-2):491–507, feb 2004.

[DPG+14] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[EvHS16] Alejandro Castedo Echeverri, Benedict von Harling, and Marco Serone. The effective bootstrap, 2016.

[FGG73] S Ferrara, A.F Grillo, and R Gatto. Tensor representations of conformal algebra and conformally covariant operator product expansion. *Annals of Physics*, 76(1):161–188, 1973.

[FK95] L.D. Faddeev and G.P. Korchemsky. High energy QCD as a completely integrable model. *Physics Letters B*, 342(1-4):311–322, jan 1995.

[FM21] Pietro Ferrero and Carlo Meneghelli. Bootstrapping the half-bps line defect cft in n=4 supersymmetric yang-mills theory at strong coupling. *Phys. Rev. D*, 104(8):L081703, 2021.

[Fuk69] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[FvHM18] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GGJ20] David Grabner, Nikolay Gromov, and Julius Julius. Excited states of one-dimensional defect cfts from the quantum spectral curve. *Journal of High Energy Physics*, 2020(7):42, 2020.

[GKLV14] Nikolay Gromov, Vladimir Kazakov, Sèbastien Leurent, and Dmytro Volin. Quantum spectral curve for planar n = 4 super-yang-mills theory. *Physical Review Letters*, 112(1), jan 2014.

[GKLV15] Nikolay Gromov, Vladimir Kazakov, Sébastien Leurent, and Dmytro Volin. Quantum spectral curve for arbitrary state/operator in AdS5/CFT4. *Journal of High Energy Physics*, 2015(9), sep 2015.

[GKP98] S.S. Gubser, I.R. Klebanov, and A.M. Polyakov. Gauge theory correlators from non-critical string theory. *Physics Letters B*, 428(1-2):105–114, may 1998.

[GRT17] Simone Giombi, Radu Roiban, and Arkady A. Tseytlin. Half-BPS wilson loop and AdS2/CFT1. *Nuclear Physics B*, 922:499–527, sep 2017.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[HTAL17a] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361. PMLR, 06–11 Aug 2017.

[HTAL17b] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies, 2017.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KNPR23] Gergely Kàntor, Vasilis Niarchos, Constantinos Papageorgakis, and Paul Richmond. 6d (2,0) bootstrap with the soft-actor-critic algorithm. *Physical Review D*, 107(2), jan 2023.

[KPN22a] Gergely Kàntor, Constantinos Papageorgakis, and Vasilis Niarchos. Solving conformal field theories with artificial intelligence. *Physical Review Letters*, 128(4), jan 2022.

[KPN22b] Gergely Kàntor, Constantinos Papageorgakis, and Vasilis Niarchos. Solving conformal field theories with artificial intelligence. *Physical Review Letters*, 128(4), jan 2022.

[KPSD14] Filip Kos, David Poland, and David Simmons-Duffin. Bootstrapping mixed correlators in the 3d ising model. *Journal of High Energy Physics*, 2014(11), nov 2014.

[LHP+16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[Lip94] L. N. Lipatov. Asymptotic behavior of multicolor qcd at high energies in connection with exactly solvable spin models. *JETP Lett.*, 59:596–599, 1994.

[LMM18] Pedro Liendo, Carlo Meneghelli, and Vladimir Mitev. Bootstrapping the half-BPS line defect. *Journal of High Energy Physics*, 2018(10), oct 2018.

[LVS22] Alessandro Laio, Uriel Luviano Valenzuela, and Marco Serone. Monte carlo approach to the conformal bootstrap. *Physical Review D*, 106(2), jul 2022.

[Mal98] Juan Maldacena. Wilson loops in large n field theories. *Physical Review Letters*, 80(22):4859–4862, jun 1998.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MNW+18] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, pages 561–577, USA, 2018. USENIX Association.

[MZ03] Joseph A Minahan and Konstantin Zarembo. The bethe-ansatz for script n = 4 super yang-mills. *Journal of High Energy Physics*, 2003(03):013–013, mar 2003.

[Nes83] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $o\left(\frac{1}{k^2}\right)$. *Dokl. Akad. Nauk SSSR,*, 269(3):543–547, 1983.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[OMK19] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. A survey of the usages of deep learning in natural language processing, 2019.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[Pol64] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[Pol74] A. M. Polyakov. Nonhamiltonian approach to conformal quantum field theory. *Zh. Eksp. Teor. Fiz.*, 66:23–42, 1974.

[Pol98] Joseph Polchinski. *String Theory*, volume 1 of *Cambridge Monographs on Mathematical Physics*. Cambridge University Press, 1998.

[PRER12] Duccio Pappadopulo, Slava Rychkov, Johnny Espin, and Riccardo Rattazzi. OPE Convergence in Conformal Field Theory. *Phys. Rev. D*, 86:105043, 2012.

[PRV19] David Poland, Slava Rychkov, and Alessandro Vichi. The conformal bootstrap: Theory, numerical techniques, and applications. *Reviews of Modern Physics*, 91(1), jan 2019.

[Qua15] Joshua D. Qualls. Lectures on conformal field theory, 2015.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[RN94] G. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

[RRSD+21] Marten Reehorst, Slava Rychkov, David Simmons-Duffin, Benoit Sirois, Ning Su, and Balt van Rees. Navigator function for the conformal bootstrap. *SciPost Physics*, 11(3), sep 2021.

[RRTV08] Riccardo Rattazzi, Vyacheslav S Rychkov, Erik Tonni, and Alessandro Vichi. Bounding scalar operator dimensions in 4d cft. *Journal of High Energy Physics*, 2008(12):031–031, dec 2008.

[Ryc17] Slava Rychkov. *EPFL Lectures on Conformal Field Theory in $D \geq 3$ Dimensions*. Springer International Publishing, 2017.

[SAC17] John Schulman, Pieter Abbeel, and Xi Chen. Equivalence between policy gradients and soft q-learning. *CoRR*, abs/1704.06440, 2017.

[SB18] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[Sch08] M. Schottenloher. *A Mathematical Introduction to Conformal Field Theory*. Lecture Notes in Physics. Springer Berlin Heidelberg, 2008.

[SD15] David Simmons-Duffin. A semidefinite program solver for the conformal bootstrap, 2015.

[SD16] David Simmons-Duffin. *TASI Lectures on the Conformal Bootstrap*. arXiv, 2016.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[SJLS00] Satinder Singh, Tommi Jaakkola, Michael Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 03 2000.

[SMG14] A Saxe, J McClelland, and S Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Represenatations 2014*. International Conference on Learning Represenatations 2014, 2014.

[Sze10] Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers, 2010.

[Wil92] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992.

[WM04] D. Wilson and Tony Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks : the official journal of the International Neural Network Society*, 16:1429–51, 01 2004.

[YLCT20] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.