Technische Universität München
T30f Theoretische Teilchen- und Kernphysik

Neural Network Parametrisation of Generalised Wilson Loops in Lattice QCD

Bachelor's Thesis
of
Verena Bellscheidt

Degree program: B. Sc. Physics

Supervisor: Prof. Dr. Nora Brambilla
Co-Supervisor: Julian Mayer-Steudte

Garching bei München, Germany
16th September 2024

Abstract

The Wilson loop is the operator commonly used in lattice QCD to compute the static potential of a heavy quark-antiquark pair. In this work, we use a neural network to optimise the spatial segments of the Wilson loop such that the overlap with either the quarkonium ground state or a specified quarkonium hybrid state is maximised. Using these optimised Wilson loops, we compute the static potentials of the quarkonium ground state and the hybrid state Π_u. For the former, our method yields later effective mass plateaus than the same computation with planar Wilson loops, corresponding to a larger overlap with the ground state. In the case of the ground state, we perform this computation for both on-axis and off-axis separations between quark and antiquark.

Zusammenfassung

Der Wilson-Loop ist der Operator, der in der Quantenchromodynamik auf dem Gitter verwendet wird, um das statische Potential eines schweren Quark-Antiquark-Paares zu bestimmen. In der vorliegenden Arbeit wird ein neuronales Netz verwendet, um die räumlichen Segmente des Wilson-Loops so zu optimieren, dass die Überlappung mit einem spezifizierten Quarkonium-Zustand, d.h. entweder dem Grundzustand oder einer exotischen, gluonischen Anregung, maximiert wird. Im Falle des Grundzustands führt diese Methode zu früheren Plateaus in der effektiven Masse, was bestätigt, dass sie die Überlappung mit dem Grundzustand erhöht.

Contents

1 Introduction **1**

2 Elements of Quantum Chromodynamics **3**
2.1 Path Integrals and Correlators . . . 3
2.2 The Quark Model . . . 4
2.3 The QCD Lagrangian . . . 4
2.4 Pure Gauge Theory . . . 6
2.5 The Static Quark-Antiquark Potential . . . 8
2.6 Hybrid States . . . 8
2.7 Running Coupling . . . 10

3 Lattice Gauge Theory **11**
3.1 Wick Rotation . . . 11
3.2 Rescaling the Gauge Fields . . . 12
3.3 The Euclidean Correlator . . . 12
3.4 Important Quantities . . . 13
3.4.1 Link Variables . . . 13
3.4.2 Plaquettes . . . 14
3.4.3 Wilson Gauge Action . . . 14
3.4.4 The Wilson loop . . . 16
3.5 Computation of Static Potentials . . . 17
3.5.1 Off-axis Separations . . . 18
3.6 Scale Setting . . . 19
3.7 Monte Carlo Integration . . . 20
3.7.1 Sampling a Set of Gauge Field Configurations . . . 20
3.8 The Continuum Limit . . . 21
3.9 Lattice Setup . . . 22

4 The Neural Network **23**
4.1 Introduction to Neural Networks . . . 23
4.1.1 Training . . . 23
4.2 Modification of the Planar Wilson Loop . . . 24
4.3 Input Data . . . 25
4.4 Layers . . . 25
4.5 The Loss Function . . . 28
4.6 Off-Axis Separations . . . 30
4.7 Limitations of the Neural Network Parametrisation . . . 30

5 Results for the Static Quark-Antiquark Potential **31**
5.1 Effective Mass and Static Potential . . . 31
5.2 Composition of the Neural Network Parametrised Superposition . . . 31

6 Application to Hybrid States **37**
6.1 Basic Concept . . . 37
6.2 Notation . . . 37
6.2.1 Action of $\mathcal{C}$, $\mathcal{P}$, $\mathcal{P}_x$ and R on a Path . . . 38
6.3 Construction of Initial States with Specified Quantum Numbers . . . 38
6.4 Propagation of Quantum Numbers through the Neural Network . . . 39
6.4.1 Action of $\mathcal{C}$, $\mathcal{P}$, $\mathcal{P}_x$ and R on the Output of a Neural Network Layer . . . 39
6.5 Layer Setup . . . 42
6.6 Results . . . 42

7 Outlook and Conclusion **45**

References **46**

A Code for the Neural Network **49**
A.1 Neural Network . . . 49

Acknowledgements **52**

1 Introduction

Quantum chromodynamics (QCD) is the quantum field theory that describes the strong interaction. Its gauge group SU(3) is part of the standard model gauge group $\mathrm{U}(1) \times \mathrm{SU}(2) \times \mathrm{SU}(3)$, making it an important pillar of our current understanding of particle physics.

A distinctive feature of low-energy QCD is a very strong coupling between colour-charged particles. This coupling confines strongly interacting particles into composite structures called hadrons. Among them are protons and neutrons, which are bound into atomic nuclei through the strong interaction. Therefore, low-energy QCD gives rise to the structure of baryonic matter as we know it.

Although a very successful model to describe hadrons was developed by Gell-Mann already in the 1960s [1], these particles are far from fully understood. For example, QCD allows for hadrons that are not included in Gell-Mann's model, referred to as exotics. On the experimental side, first evidence for hadrons beyond the quark model was found by the Belle collaboration in 2003 [2]. Since then, many more so-called XYZ states, which are candidates for exotics, have been discovered [3]. This has led to an increased interest in exotics in recent years.

Another consequence of the strong coupling at low energies is that perturbative computations fail in this regime. This necessitates a non-perturbative treatment of QCD, which can be achieved within the well-established framework of lattice gauge theory. Lattice computations have e.g. been used to compute the static potentials of quarkonium and quarkonium-related exotics [4, 5], which in turn yield information on the kinematically allowed decay channels of these particles.

In this work, we optimise the lattice computation of the potentials of a static quark-antiquark pair and of hybrid quarkonia, which are exotic mesons. Our computation makes use of so-called Wilson loops, which should ideally have a large overlap with the state under consideration. Contrary to previous approaches, we employ a neural network to optimise the ground state overlap of the Wilson loop. We compare our results to the ones obtained with non-optimised Wilson loops and discuss potential future extensions of our method.

This thesis is structured as follows: We review relevant results from QCD in Chapter 2, followed by a discussion of lattice QCD in Chapter 3. This lays the foundations for our neural network parameterization, which we describe in detail in Chapter 4. The results for the quarkonium ground state are presented in Chapter 5, while Chapter 6 provides the generalisation to arbitrary hybrid states. Finally, in Chapter 7, we summarise our results and discuss perspectives for future research.

2 Elements of Quantum Chromodynamics

The main objective of this work is to develop a neural network to find a lattice interpolator with maximised overlap with a specified state of a static quark-antiquark pair. In this chapter, we review the theory underlying this problem: quantum chromodynamics (QCD).

2.1 Path Integrals and Correlators

QCD is the quantum field theory (QFT) that describes the strong interaction. The most common formulation of QFT is based on Richard Feynman's path integral formulation of quantum mechanics [6]. In this formulation of quantum mechanics, one uses the fact that the transition amplitude for propagation from $\langle \mathbf{x} | \psi(t) \rangle$ at time t to $\langle \mathbf{x'} | \psi(t') \rangle$ at time t' is given by

$$\left\langle \mathbf{x'} \,\middle|\, \psi(t') \right\rangle = \int \mathrm{d}\mathbf{x}\, G(\mathbf{x'}, t'; \mathbf{x}, t)\, \langle \mathbf{x} \,|\, \psi(t) \rangle \tag{2.1}$$

where[1]

$$G(\mathbf{x'}, t'; \mathbf{x}, t) = \left\langle \mathbf{x'} \,\middle|\, e^{-iH(t'-t)} \,\middle|\, \mathbf{x} \right\rangle = \int \mathcal{D}\mathbf{x}\; e^{iS[\mathbf{x}]} \tag{2.2}$$

is the Green's function for the Schrödinger equation with boundary condition $G(\mathbf{x}, t; \mathbf{x'}, t_0) = 0$ for $t < t_0$. The last equality is derived for a non-relativistic one-particle Hamiltonian e.g. in [7]. The path integral runs over all paths that satisfy $\mathbf{x}(t) = \mathbf{x}$ and $\mathbf{x}(t') = \mathbf{x'}$. $S[\mathbf{x}]$ denotes the classical action associated with path $\mathbf{x}(t)$.

Similarly, for a complex fermionic quantum field ψ that couples to a gauge field A_μ, the correlator is given by

$$\langle \Omega \,|\, \mathcal{T}\{O_2(x_2)\, O_1(x_1)\} \,|\, \Omega \rangle = \frac{1}{Z} \int \mathcal{D}\bar{\psi}\, \mathcal{D}\psi\, \mathcal{D}A\; O_2(x_2)\, O_1(x_1)\, e^{iS_{\mathrm{QCD}}} \tag{2.3}$$

$$Z = \int \mathcal{D}\bar{\psi}\, \mathcal{D}\psi\, \mathcal{D}A\; e^{iS_{\mathrm{QCD}}} \tag{2.4}$$

where $|\Omega\rangle$ is the vacuum state, $\mathcal{T}$ is the time ordering prescription, and O_1 and O_2 are products of fermion fields and gauge fields. Such correlators are important in QFT as they are propagation amplitudes and can be used to calculate expectation values of observables. We will use a correlator to obtain the static quark-antiquark potential, but only after going to imaginary time, as will be explained in Section 3.1.

From (2.3) and (2.4) one can see that the QCD action S_{QCD}, which is the spacetime integral of the QCD Lagrangian density $\mathcal{L}_{\mathrm{QCD}}$, determines the contributions of a field configuration $(\psi, \bar{\psi}, A)$ to the propagation amplitude. Finding a discretised version of the action was central to establishing a lattice gauge theory of the strong interaction, as will become apparent in Chapter 3. To find such a discretised action, we first need to discuss its continuum counterpart. This, in turn, requires us to know the QCD Lagrangian.

[1] We use natural units: $c = \hbar = k_{\mathrm{B}} = 1$.

2.2 The Quark Model

Before constructing the QCD Lagrangian, we discuss the quark model in order to understand the symmetry properties of this Lagrangian as well as the importance of hybrid states. Historically, the quark model explained the so-called "particle zoo", which consisted of the numerous then-known strongly interacting particles. In the 1960s, Gell-Mann [1] and Zweig [8] independently proposed that these particles are not fundamental but consist of the spin-1/2 particles that are now called quarks. Their initial model predicts three distinct quark flavours: up (u), down (d), and strange (s). Discoveries of further hadrons revealed three more flavours: charm (c), bottom (b), and top (t). The spectrum of baryons, which are compound particles with an odd number of valence quarks, required the spin and flavour quantum numbers to be totally symmetric under exchange of particles[2]. As baryons have half-integer spin, this contradicts the spin-statistics theorem, which requires the overall wave function to be totally antisymmetric. This was resolved by Greenberg [9] and Han and Nambu [10, 11], who proposed the existence of an unobserved quantum number called *colour* in which baryon wave functions are totally antisymmetric. Colour underlies an SU(3) symmetry, often denoted as $\mathrm{SU}(3)_\mathrm{C}$. This SU(3) symmetry group is the defining characteristic of QCD.

(Anti)quarks transform under the fundamental (anti)triplet representation of SU(3). The strong interaction confines them into hadrons, which are $\mathrm{SU}(3)_\mathrm{C}$ singlets. Gell-Mann's quark model describes those hadrons whose quantum numbers are determined by either a quark and an antiquark ($q\bar{q}$) or three (anti-)quarks (qqq or $\bar{q}\bar{q}\bar{q}$).

The quark model does, however, not encompass all possible hadrons. So-called *exotic hadrons* are allowed in QCD but not included in the quark model. On the experimental side, so-called XYZ states, which are not predicted by the quark model, have been observed. We will return to this discussion in Section 2.6.

2.3 The QCD Lagrangian

As quarks are spin-1/2 particles of different flavours and colours, they can be described by Dirac spinors ψ that carry a flavour quantum number $f = 1, 2, ..., N_\mathrm{f}$ and a colour index $i = 1, 2, 3$ (red, green, blue). Therefore, we start the construction of the QCD Lagrangian from the Lagrangian of a free fermion,

$$\mathcal{L}_\mathrm{Dirac} = \overline{\psi}\left(i\partial\!\!\!/ - m\right)\psi \tag{2.5}$$

where summations over the flavour quantum numbers and colour indices are implicit.

Following [12] and [13] as well as our discussions above, we require the QCD Lagrangian to be invariant under colour gauge transformations $\psi \rightarrow \Omega\,\psi$ ($\Omega \in \mathrm{SU}(3)$). Parameterising Ω as $\Omega = \exp(i\sum \epsilon_a T_a)$ where the $T_a = \frac{1}{2}\lambda_a$ are half the Gell-Mann matrices, which are the generators of SU(3), we see that the free Dirac Lagranigan (2.5) is invariant under global SU(3) transformations. For local SU(3) transformations, i.e. if the parameters $\epsilon_a(x)$ depend on the spacetime position x, the mass term in (2.5) is still invariant. However, the kinetic term contains a partial derivative, which is defined by

$$n^\mu \partial_\mu \psi(x) = \lim_{\epsilon\to 0} \frac{1}{\epsilon}\left[\psi(x+\epsilon n) - \psi(x)\right]. \tag{2.6}$$

As $\psi(x)$ and $\psi(x+\epsilon n)$ transform differently under local transformations, the kinetic term in (2.5) is not invariant under local colour gauge transformations.

In order to resolve this problem, we introduce the *link variable* $U(y,x) \in \mathrm{SU}(3)$ which satisfies

$$U(y,x) \rightarrow \Omega(y)\,U(y,x)\,\Omega^\dagger(x), \qquad U(x,x) = 1. \tag{2.7}$$

[2]Consider e.g. the Δ^{++} baryon: it is a uuu bound state where all quark spins are parallel.

The link variable can be used to define the covariant derivative D_μ by

$$n^\mu D_\mu \psi(x) = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[\psi(x + \epsilon n) - U(x + \epsilon n, x)\, \psi(x) \right] \tag{2.8}$$

which transforms into

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[\Omega(x + \epsilon n)\, \psi(x + \epsilon n) - \Omega(x + \epsilon n)\, U(x + \epsilon n, x)\, \Omega^\dagger(x)\, \Omega(x)\, \psi(x) \Big] \tag{2.9}$$

$$= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \Omega(x + \epsilon n) \left[\psi(x + \epsilon n) - U(x + \epsilon n, x)\, \psi(x) \right] \tag{2.10}$$

$$= \Omega(x)\, n^\mu D_\mu \psi(x) \tag{2.11}$$

As n^μ is an arbitrary 4-vector, it follows that

$$D_\mu \psi(x) \;\to\; \Omega(x)\, D_\mu \psi(x) \tag{2.12}$$

i.e. the covariant derivative of ψ transforms in the same way as ψ itself. Hence, in order to obtain a locally invariant Lagrangian from one that is only globally invariant, one has to replace the derivatives with covariant derivatives. In the case of the kinetic term of the Dirac Lagrangian, one replaces $\bar\psi \not\partial \psi$ with $\bar\psi(x) \not D \psi(x)$, which is invariant under local gauge transformations:

$$\bar\psi(x) \not D\, \psi(x) \;\to\; \bar\psi(x)\, \Omega^\dagger(x)\, \Omega(x)\, \not D\, \psi(x) \;=\; \bar\psi(x) \not D\, \psi(x). \tag{2.13}$$

Since $U(y, x) \in \mathrm{SU}(3)$, we can expand $U(x + \epsilon n, x)$ as

$$U(x + \epsilon n, x) = 1 + \epsilon\, n^\mu i g A_\mu(x) + \mathcal{O}(\epsilon^2) \tag{2.14}$$

where the vector field A_μ is an element of the Lie algebra $\mathfrak{su}(3)$, and therefore can be written as $A_\mu(x) = \Sigma A_{a\mu}(x)\, T_a$. In this way, local colour gauge invariance implies the existence of eight vector fields A^μ_a in the adjoint (octet) representation of SU(3). The bosons described by these vector fields are gluons, and the index a corresponds to the adjoint colour that a gluon carries. Whether or not the coupling parameter g is written in this expansion is a matter of convention; it can equally well be absorbed into A_μ.

To describe the dynamics of the gluons, we insert (2.14) into the definition of the covariant derivative (2.8), and find that

$$D_\mu = \partial_\mu - i g A_\mu. \tag{2.15}$$

This allows us to write down the gluon field strength tensor as

$$F_{\mu\nu} = \frac{i}{g} \left[D_\mu, D_\nu \right] = \partial_\mu A_\nu - \partial_\nu A_\mu - i g \left[A_\mu, A_\nu \right] = \partial_\mu A_\nu - \partial_\nu A_\mu - i g A_{a\mu} A_{b\nu} \left[T_a, T_b \right] = \Sigma_a F_{a\mu\nu} T_a \tag{2.16}$$

where the last equality holds because $F_{\mu\nu}$, like A_μ, is an element of the Lie algebra $\mathfrak{su}(3)$. As SU(3) is non-abelian, its generators do not commute, and the commutator in (2.16) does not vanish. As a consequence, the squared field strength tensor $F_{a\mu\nu} F^{\mu\nu}_a$ contains cubic and quartic terms in A_μ. Since $F_{a\mu\nu} F^{\mu\nu}_a$ appears in the QCD Lagrangian, this means that there are three-gluon vertices as well as four-gluon vertices. These are depicted in Figure 2.1.

Finally, we write down the QCD Lagrangian as the Yang-Mills Lagrangian for SU(3):

$$\mathcal{L}_{\mathrm{QCD}} = \bar\psi_i \left(i \gamma^\mu (D_\mu)_{ij} - m\, \delta_{ij} \right) \psi_j - \frac{1}{4} F_{a\mu\nu} F^{\mu\nu}_a \tag{2.17}$$

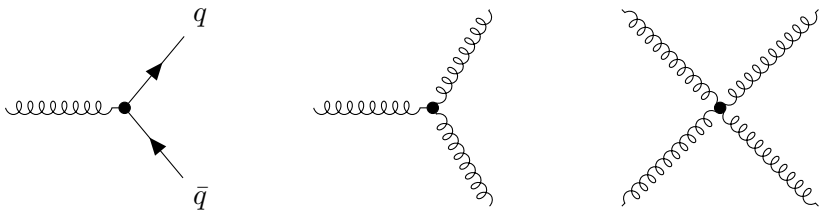


Figure 2.1: In QCD there is a $q\bar{q}g$ vertex (left-hand-side Feynman diagram) that resembles the $e^-e^+\gamma$ vertex from QED, as well as a three- and four-gluon vertices (middle and right-hand-side Feynman diagrams).

2.4 Pure Gauge Theory

To determine the transformation behaviour of the gauge fields, we require $D_\mu\psi$ to transform according to (2.12) and insert (2.15):

$$D_\mu\psi \;\to\; D'_\mu\psi' \;=\; \Omega\, D_\mu\psi \tag{2.18}$$

$$\Rightarrow (\partial'_\mu - igA'_\mu)\,\psi' \;=\; \Omega\,(\partial_\mu - igA_\mu)(\Omega^\dagger\psi') \tag{2.19}$$

$$\Rightarrow\; -igA'_\mu\psi' \;=\; \Omega\,\partial_\mu(\Omega^\dagger\psi') - \partial'_\mu\psi' - ig\,\Omega\,A_\mu\Omega^\dagger\psi' \tag{2.20}$$

$$\Rightarrow\; -igA'_\mu\psi' \;=\; \Omega\,(\partial_\mu\Omega^\dagger)\psi' - ig\,\Omega\,A_\mu\Omega^\dagger\psi' \tag{2.21}$$

$$\Rightarrow A'_\mu \;=\; \Omega\,A_\mu\,\Omega^\dagger + \frac{i}{g}\,\Omega\,\partial_\mu\,\Omega^\dagger. \tag{2.22}$$

In this work, we work exclusively in pure gauge, meaning that we neglect the dynamics of the fermions. To understand what this means in practice, consider the correlator (2.3) written in terms of a fermionic part and a gauge field part:

$$\langle O\rangle \;=\; \frac{1}{Z}\int \mathcal{D}A\; e^{iS_\mathrm{G}[A]}\, Z_\mathrm{F}[A]\, \frac{1}{Z_\mathrm{F}[A]}\int \mathcal{D}\bar{\psi}\,\mathcal{D}\psi\; e^{iS_\mathrm{F}[\psi,\bar{\psi},A]}\, O[\psi,\bar{\psi},A] \;\equiv\; \langle\langle O\rangle_\mathrm{F}\rangle_\mathrm{G} \tag{2.23}$$

where we defined

$$\langle O\rangle_\mathrm{F} \;=\; \frac{1}{Z_\mathrm{F}[A]}\int \mathcal{D}\bar{\psi}\,\mathcal{D}\psi\; e^{iS_\mathrm{F}[\psi,\bar{\psi},A]}\, O[\psi,\bar{\psi},A] \tag{2.24}$$

and

$$\langle O\rangle_\mathrm{G} \;=\; \frac{1}{Z}\int \mathcal{D}A\; e^{iS_\mathrm{G}[A]}\, Z_\mathrm{F}[A]\, O[A]. \tag{2.25}$$

The quantity

$$Z_\mathrm{F}[A] \;=\; \int \mathcal{D}\bar{\psi}\,\mathcal{D}\psi\;\; e^{iS_\mathrm{F}[\psi,\bar{\psi},A]} \tag{2.26}$$

is called the fermionic partition function, or *fermion determinant*.

When working in pure gauge, one sets $\langle O\rangle = \langle O\rangle_\mathrm{G}$, thereby neglecting the dynamics of the fermions. This corresponds to the limit of infinitely heavy quarks, where the fermion determinant approaches unity. Hence, the correlator reduces to

$$\langle O\rangle_{\text{pure gauge}} \;=\; \frac{1}{Z_\mathrm{G}}\int \mathcal{D}A\; e^{iS_\mathrm{G}[A]}\, O[A] \tag{2.27}$$

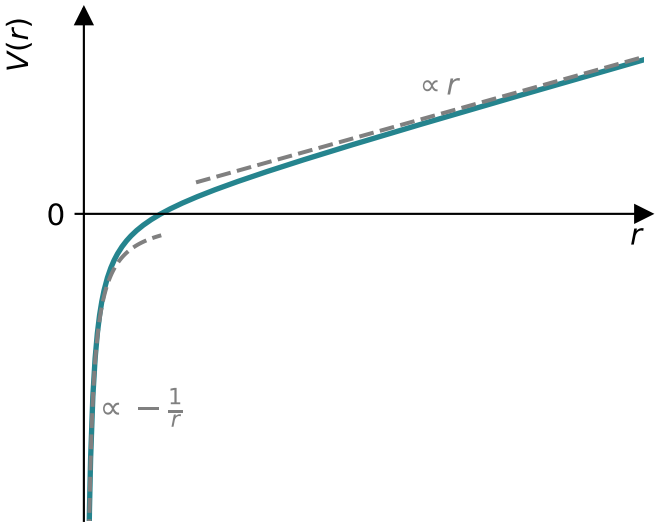


Figure 2.2: At small separations r, the Cornell potential (turquoise curve) behaves as $1/r$, while at large separations it approaches a linear potential.

where $Z_{\mathrm{G}} \equiv Z \cdot \langle 1 \rangle_{\mathrm{G}}$ approximates the partition function in the same in which (2.27) approximates the correlator. Setting the fermion determinant to unity is referred to as the *quenched approximation*.

For the computation of correlators, this means that we replace the QCD action with only the gauge action, which can be read off of (2.17) as

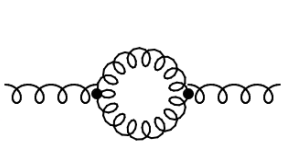
$$S_{\mathrm{G}}[A] = -\frac{1}{4}\int \mathrm{d}^4x \; F_{a\mu\nu}(x)F_a^{\mu\nu}(x) \tag{2.28}$$

$$= -\frac{1}{4}\int \mathrm{d}^4x \, F_{a\mu\nu}(x)F_b^{\mu\nu}(x)\,\frac{1}{2}\underbrace{\mathrm{tr}[\,\lambda_a\lambda_b\,]}_{2\,\delta_{ab}} \tag{2.29}$$

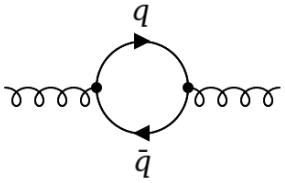
$$= -\frac{1}{2}\int \mathrm{d}^4x \; \mathrm{tr}\left[\frac{1}{2}\,F_{a\mu\nu}(x)\,\lambda_a\,\frac{1}{2}\,F_b^{\mu\nu}(x)\,\lambda_b\right] \tag{2.30}$$

$$= -\frac{1}{2}\int \mathrm{d}^4x \; \mathrm{tr}[F_{\mu\nu}(x)F^{\mu\nu}(x)] \tag{2.31}$$

In conclusion, pure gauge simplifies the expressions for the QCD action and the correlator under the assumption of infinitely heavy quarks. From here on, we work only in pure gauge.



Possible in pure gauge



Not included in pure gauge

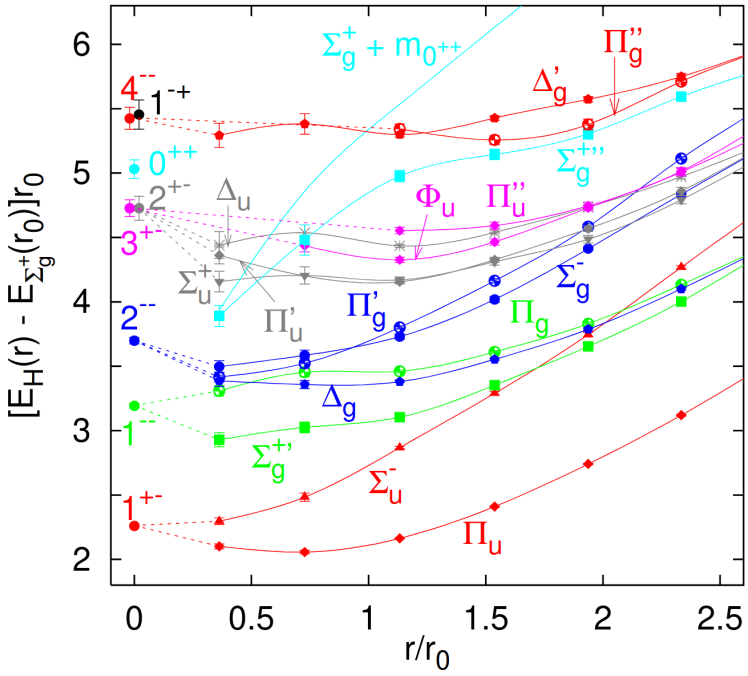


Figure 2.3: Static potentials of quarkonium hybrid states. $r_0 \approx 0.5\,\text{fm}$ is the Sommer scale. Figure taken from [16].

2.5 The Static Quark-Antiquark Potential

The physical system central to this work is a bound state of a heavy quark Q and an antiquark $\bar{Q}$ of the same flavour. Such a state is referred to as *quarkonium*. As the top quark decays too quickly to form bound states, the quark flavour is either charm or bottom.

In the 1970s, the so-called Cornell potential was proposed to describe the ground state energy of quarkonium [14]:

$$V(r) = -\frac{A}{r} + \sigma r + \text{const.} \tag{2.32}$$

where $A = \frac{4}{3}\,\alpha_S\,\hbar c$. σ is the so-called *string tension*. While originally motivated by phenomenology, this potential can also be obtained from lattice calculations, as will be discussed in detail in Section 3.5. For a discussion of the Cornell potential in the context of potential non-relativistic QCD (pNRQCD), we refer to [15].

The form of the Cornell potential is shown in Figure 2.2. At small distances r, the potential behaves as r^{-1} coming from a weak coupling expansion and corresponding to a Coulomb potential. For large values of r, a strong coupling expansion yields a potential linear in r. If energy is added to the system, this will not separate the quark-antiquark pair into a quark and an antiquark. Instead, the energy at some point suffices to create another hadron. This reflects the fact that at low energies, colour-charged particles are confined in hadrons and not observed in isolation. This phenomenon is called confinement.

2.6 Hybrid States

Gell-Mann's quark model does not account for all states that are allowed in QCD. Hadron states not included in the quark model are called *exotic hadrons* or *exotics*. Among them are so-called hybrid states where gluons contribute to the quantum numbers. If only gluons determine the quantum numbers, the state is called a glueball. Multiquark states, such as tetraquarks and pentaquarks, are also exotic.

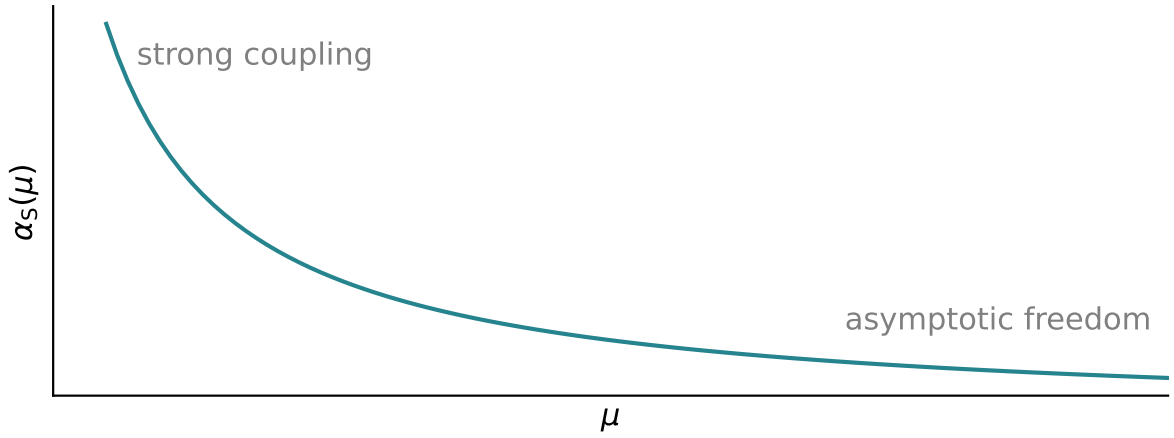


Figure 2.4: The strong coupling constant α_S is a decreasing function of the energy scale μ: for large values of μ, the coupling is weak, and the particles are *asymptotically free*, allowing for a perturbative treatment of high energy QCD problems. For small values of μ, the strong coupling diverges, leading to the failure of perturbative approaches.

Exotics are subject to ongoing research as they provide a way to experimentally test QCD at low energies. In 2003, a potentially exotic state was observed for the first time at the Belle experiment [2]. Since then, several further candidates have been discovered [3]. They are collectively known as *XYZ states*. Some of them, such as charged charmonium-like states, clearly are exotic, as their quantum numbers are not accessible to non-exotic mesons. For a more detailed treatment of XYZ states, we refer to [17].

The method we develop in this work is designed to investigate hybrid quarkonia, whose valence content consists of a heavy quark, an antiquark of the same flavour, and a number of gluons. They are labelled by three quantum numbers Λ_η^ϵ which correspond to the irreducible representations of the infinite group of cylindrical rotations with reflections $D_{\infty h}$. $\Lambda \in \{\, \Sigma(=0), \Pi(=1), \Delta(=2), ... \,\}$ is the angular momentum along the quark-antiquark separation axis, $\eta \in \{\, g(=1), u(=-1) \,\}$ describes the parity under $\mathcal{CP}$ transformations, and $\epsilon \in \{\, +, - \,\}$ is the eigenvalue of the reflection along an axis perpendicular to the quark-antiquark separation axis. For $\Lambda > 0$, the irreducible representation Λ_η is two-dimensional and the states are degenerate in $\epsilon = \pm 1$. In these cases, the ϵ-label is omitted. For example, the non-exotic quarkonium ground state is labelled Σ_g^+, and the first gluonic excitation by Π_u. Lattice data for the static potentials of these two states is shown in Figure 2.3.

Lattice data has also been used to parametrise the static potentials [4, 5]. In particular, in the range $2a \leq r$ ($a = 0.0571(4)\,\mathrm{fm}$) Schlosser and Wagner [5] found that the Σ_u^- and Π_u potentials can be described by the fit function

$$V_{\Lambda_\eta^\epsilon}(r) \;=\; \frac{A_1}{r} + A_2 + A_{3,\Lambda_\eta^\epsilon}\, r^2. \tag{2.33}$$

While the coefficients A_1 and A_2 are equal for the Σ_u^- and Π_u potentials, the coefficients of the quadratic term, $A_{3,\Lambda_\eta^\epsilon}$, are independent of each other.

The static potentials can help identify kinematically allowed decay channels of hybrid mesons. Of particular interest are hybrid states with $J^{\mathcal{PC}}$ quantum numbers such as 0^{--} or 0^{+-} that are not accessible to non-exotic mesons as these states may produce distinctive final states, facilitating their experimental identification. The $J^{\mathcal{PC}}$ quantum numbers describe the $r \to 0$ limit of the hybrid states, and the latter are degenerate in this limit. For example Σ_u^- and Π_u both correspond to $J^{\mathcal{PC}} = 1^{+-}$, where Σ_u^- corresponds to the $J_z = 0$ state and Π_u to the $J_z = \pm 1$ doublet.

2.7 Running Coupling

In general, coupling parameters g depend on the energy scale μ of the process under consideration. This phenomenon is referred to as *running coupling*.

In the case of QCD at sufficiently large energies, the strong coupling constant α_S is given by

$$\alpha_S = \frac{g^2}{4\pi} \approx \frac{1}{\beta_0 \ln\left(\mu^2/\Lambda^2_{\mathrm{QCD}}\right)} \tag{2.34}$$

where β_0 is a constant and Λ_{QCD} is the QCD scale. Note that the above expression does not hold at energies μ below or at the pole $\mu = \Lambda_{\mathrm{QCD}}$. For $\mu > \Lambda_{\mathrm{QCD}}$, the functional form of $\alpha_S(\mu)$ is plotted in Figure 2.4.

The running of α_S gives rise to two characteristic features of QCD:

- *Asympotitc freedom*: At large values of μ, i.e. in the high energy regime, the strong coupling constant tends to zero. In this regime, one can expand the exponential in (2.3) in powers of the coupling constant. Wick's theorem than leads to a sum of Feynman diagrams.
- *Confinement*: For small values of μ, the strong coupling constant reaches the order of unity, rendering an expansion for small coupling constants inexpedient. This means that perturbative results such as (2.34) do not hold at small energies. In this regime, the strong interaction confines colour-charged particles into hadrons, which are colour singlets.

We see that perturbative approaches fail in low-energy QCD. A non-perturbative treatment of QCD is therefore needed.

3 Lattice Gauge Theory

A well-established non-perturbative approach to QCD is lattice QCD. Instead of interpreting the fields as virtual particles and adding the contributions of the leading Feynman diagrams, one numerically simulates the quantum fields. In order to render this task numerically feasible, the fields are simulated on a discrete spacetime lattice

$$\Lambda = \{ (n_1, n_2, n_3, n_4) \mid n_1, n_2, n_3 = 0, 1, ..., N-1 \wedge n_4 = 0, 1, ..., N_\mathrm{T} - 1 \} \tag{3.1}$$

As will be explained in Section 3.1, we work in Euclidean time, which is why the time dimension is labelled by the index 4 instead of 0. We impose periodic boundary conditions for both temporal and spatial dimensions. In the following, we will suppress the lattice constant a and write $\psi(n) \equiv \psi(an)$. Our treatment will largely follow that of Gattringer and Lang [18].

The quarks are described by the values of the fermion field at the lattice sites, i.e. by the $\psi(n)$ and $\bar{\psi}(n)$, where $n \in \Lambda$. The bosonic degrees of freedom are contained in the link variables from Section 2.3, which correspond to links between adjacent lattice sites.

In this chapter, we discuss the lattice gauge analogues of the quantities introduced in Chapter 2. Building on this knowledge, we discuss the Wilson loop computation that is commonly done on the lattice in order to obtain the static quark-antiquark potential. This lays the foundations for our neural network-based parametrisation of the Wilson loop.

3.1 Wick Rotation

In order to calculate correlators on the lattice one has to evaluate the path integral (2.27). However, the complex phase of the integrand renders a numerical evaluation difficult. This problem can be circumvented by performing a Wick rotation.

Comparing the infinitesimal space-time interval in Minkowski space

$$ds^2 = -dt^2 + dx^2 + dy^2 + dz^2 \tag{3.2}$$

to the one in four-dimensional Euclidean space

$$(ds_E)^2 = (dt_E)^2 + dx^2 + dy^2 + dz^2, \tag{3.3}$$

it is apparent that by analytically continuing to imaginary values of time, i.e. to $t \in i\mathbb{R}$, problems in Minkowski space may be related to problems in Euclidean space.

This motivates the Wick rotation,

$$t \;\rightarrow\; -i\, t_\mathrm{E} \tag{3.4}$$

which means that the temporal integral of the action now runs over the imaginary axis. The complex phase in the path integral than turns into an exponentially decaying function:

$$i\, S \;\rightarrow\; -S_\mathrm{E} \tag{3.5}$$

As we will see, this allows us to apply the well-developed methods of statistical mechanics to QCD. For this reason, we work with a four-dimensional Euclidean space instead of a Minkowski space. We do not distinguish between co- and contravariant coordinates and use only lower indices, such as x_μ.

For a more rigorous treatment of the Wick rotation we refer to [19].

3.2 Rescaling the Gauge Fields

We now rescale the gauge fields as $gA_\mu \to A_\mu$, meaning that the gluon field strength tensor is given by $F_{\mu\nu} = \frac{1}{g}(\partial_\mu A_\nu - \partial_\nu A_\mu - i\,[A_\mu, A_\nu])$. Rescaling this according to $gF_{\mu\nu} \to F_{\mu\nu}$ entails a change of the continuum gauge action (2.31) into

$$S_{\mathrm{G}}[A] = \frac{1}{2g^2} \int \mathrm{d}^4x \ \mathrm{tr}\big[F_{\mu\nu}^2(x)\big]. \tag{3.6}$$

3.3 The Euclidean Correlator

In Euclidean space, the pure gauge correlator (2.27) is replaced by

$$\langle O_2(t_2)\, O_1(t_1)\rangle = \frac{\int \mathcal{D}U \ O_2(t_2)\, O_1(t_1)\, e^{-S_{\mathrm{E}}[U]}}{\int \mathcal{D}U \ e^{-S_{\mathrm{E}}[U]}} \tag{3.7}$$

where we suppressed the spatial arguments of $O(t_1)$ and $O(t_2)$ in our notation, and integrated over the group elements $U \in \mathrm{SU}(3)$ instead of the elements $A \in \mathfrak{su}(3)$ of the Lie algebra. Recognising $\exp(-S_{\mathrm{E}})$ as a Boltzmann factor, we are able to apply methods of statistical mechanics to QCD. As (3.7) takes the form of the expectation value of $O_2(t_2)O_1(t_1)$ in a canonical ensemble, we write it in terms of Hilbert space operators as

$$\langle O_2(t)\, O_1(0)\rangle_T = \frac{1}{Z_T} \mathrm{tr}\left[e^{-(T-t)H} O_2\, e^{-tH} O_1\right] \tag{3.8}$$

$$Z_T = \mathrm{tr}\left[e^{-TH}\right] \tag{3.9}$$

where on the right-hand side O_1, O_2, and the Hamiltonian H are Hilbert space operators. Through comparison with the Boltzmann factor $\exp(-\beta H)$ from statistical mechanics we can relate T to an inverse temperature: $T = \beta = (T_{\text{temperature}})^{-1}$. One can show that this, indeed, is the physical temperature. We will take T to be infinite, corresponding to zero temperature.

We assume a discrete energy spectrum $\sigma(H) = \{E_n\}$, where E_n is the energy of state $|n\rangle$ relative to the vacuum, i.e. $H|n\rangle = E_n|n\rangle$ $(E_n \in \sigma(H))$ and where $(n > m) \Rightarrow (E_n \geq E_m)$. We take the $T \to \infty$ limit:

$$\langle O_2(t)\, O_1(0)\rangle_T = \frac{1}{Z_T} \sum_{m,n} \langle m|\, e^{-(T-t)H} O_2\, |n\rangle\, \langle n|\, e^{-tH} O_1\, |m\rangle \tag{3.10}$$

$$= \frac{1}{Z_T} \sum_{m,n} \langle m|\, O_2\, |n\rangle\, \langle n|\, O_1\, |m\rangle\, e^{-(T-t)E_m} e^{-tE_n} \tag{3.11}$$

$$= \frac{\sum_{m,n} \langle m|\, O_2\, |n\rangle\, \langle n|\, O_1\, |m\rangle\, e^{-(T-t)E_m}\, e^{-tE_n}}{\sum_m e^{-TE_m}} \tag{3.12}$$

$$= \frac{\sum_{m,n} \langle m|\, O_2\, |n\rangle\, \langle n|\, O_1\, |m\rangle\, e^{-(T-t)E_m}\, e^{-tE_n}}{1 + \sum_{|m\rangle \neq |\Omega\rangle} e^{-TE_m}} \tag{3.13}$$

$$\xrightarrow{T\to\infty} \sum_n \langle \Omega|\, O_2\, |n\rangle\langle n|\, O_1\, |\Omega\rangle\, e^{-tE_n} \ \equiv\ \langle O_2(t)\, O_1(0)\rangle \tag{3.14}$$

Note that the large-t behaviour of the Euclidean correlator is determined solely by the ground

state:

$$\langle O_2(t)\, O_1(0)\rangle \;=\; \sum_n \langle\Omega|\, O_2\, |n\rangle\, \langle n|\, O_1\, |\Omega\rangle\, e^{-tE_n} \tag{3.15}$$

$$=\; \langle\Omega|\, O_2\, |0\rangle\, \langle 0|\, O_1\, |\Omega\rangle\; e^{-tE_0} + \sum_{n>0} \langle\Omega|\, O_2\, |n\rangle\, \langle n|\, O_1\, |\Omega\rangle\, e^{-tE_n} \tag{3.16}$$

$$\xrightarrow{t\to\infty}\; \langle\Omega|\, O_2\, |0\rangle\, \langle 0|\, O_1\, |\Omega\rangle \tag{3.17}$$

Setting $O_2 = O_1^\dagger$, we can take advantage of this behaviour to find the ground state expectation value of an operator O_1:

$$\langle O_1^\dagger(t)\, O_1(0)\rangle \;\xrightarrow{t\to\infty}\; |\, \langle\Omega|\, O_1\, |\Omega\rangle\, |^2 \tag{3.18}$$

3.4 Important Quantities

3.4.1 Link Variables

On the lattice, a local gauge transformation can be written as

$$\psi(n) \;\to\; \Omega(n)\,\psi(n), \qquad \bar\psi(n) \;\to\; \bar\psi(n)\,\Omega^\dagger(n) \qquad (\Omega \in \mathrm{SU}(3)) \tag{3.19}$$

The link variable $U_\mu(n) \in \mathrm{SU}(3)$ from Section 2.3 transforms according to

$$U_\mu(n) \;\to\; \Omega(n)\, U_\mu(n)\, \Omega^\dagger(n+\hat\mu), \tag{3.20}$$

making the product $\bar\psi(n)\, U_\mu(n)\, \psi(n+\hat\mu)$ gauge invariant. The name "link variable" now is particularly apt: $U_\mu(n)$ depends on a location n in the lattice and a direction $\hat\mu \equiv \hat e_\mu$, and can be interpreted as a link between the lattice sites n and $n+\hat\mu$. Since U is unitary, $U_\mu^\dagger(n) = U_\mu^{-1}(n) = U_{-\mu}(n+\hat\mu)$ is the link in the opposite direction. This is illustrated in Figure 3.1.

In the continuum, there exists a quantity that transforms in a manner analogous to $U_\mu(n)$: the gauge transporter

$$G(x,y) = \mathcal{P}\exp\!\left(i\int_{\mathcal{C}_{xy}} A\cdot \mathrm{d}s\right) \tag{3.21}$$

where $\mathcal{P}$ denotes path ordering, and the integral is along a curve $\mathcal{C}_{xy}$ that connects x and y. The gauge transporter transforms according to $G(x,y) \to \Omega(x)\, G(x,y)\, \Omega^\dagger(y)$, which is why we interpret the link variable $U_\mu(n)$ as the lattice counterpart of the gauge transporter. Taking this analogy further, we write

$$U_\mu(n) = \exp\big(iaA_\mu(n)\big) \tag{3.22}$$

where $A_\mu(n) \in \mathrm{su}(3)$ is a lattice gauge field. The exponent in this expression approximates the one in (3.21) to first order in a, which is why no path ordering is necessary. In the continuum limit $a \to 0$ the two expressions agree.

In Section 2.3 we introduced gauge bosons in order to account for the possibility of different gauge transformations of the fermion field at different spacetime points. Intuitively, it makes sense that on the lattice the bosonic degrees of freedom are contained in the links between adjacent lattice sites.

Figure 3.1: Visual interpretation of the link variables: $U_\mu(n)$ is the link from n to $n+\hat{\mu}$, and $U_\mu^\dagger(n)$ the one from $n+\hat{\mu}$ to n.

3.4.2 Plaquettes

In order to extract meaningful information from a set of lattice gauge configurations one needs to construct functionals of the lattice fields that correspond to physical observables. In particular, these functionals must be gauge invariant. In pure gauge theory, this implies that the terms of such a functional are a products of link variables that corresponds to a path $\mathcal{P}$ on the lattice. These functionals are referred to as *interpolators* or *operators*, and can in pure gauge be identified with their corresponding paths $\mathcal{P}$. The evaluation of an observable on the lattice is commonly referred to as *measurement*.

For the sake of more convenient notation we define

$$\prod_{(n,\,\mu)\in\mathcal{P}} U_\mu(n) \;\equiv\; U_{\mu_0}(n_0)\, U_{\mu_1}(n_0+\hat{\mu}_0)\,...\,U_{\mu_{k-1}}(n_k-\hat{\mu}_{k-1}) \tag{3.23}$$

whose behaviour under gauge transformations is given by

$$\begin{aligned}\prod_{(n,\,\mu)\in\mathcal{P}} U_\mu(n) \;\to\; &\Omega(n_0)\, U_{\mu_0}(n_0)\, \Omega^\dagger(n_0+\hat{\mu}_0)\, \Omega(n_0+\hat{\mu}_0)\, U_{\mu_1}(n_0+\hat{\mu}_0)\, \Omega^\dagger(n_0+\hat{\mu}_0+\hat{\mu}_1)\,... \\ &...\,\Omega(n_k-\hat{\mu}_{k-1})\, U_{\mu_{k-1}}(n_k-\hat{\mu}_{k-1})\, \Omega^\dagger(n_k)\end{aligned} \tag{3.24}$$

$$= \;\Omega(n_0)\left(\prod_{(n,\,\mu)\in\mathcal{P}} U_\mu(n)\right)\Omega^\dagger(n_k). \tag{3.25}$$

Replacing the general path $\mathcal{P}$ by a closed loop $\mathcal{L}$ and taking the trace, we construct a gauge invariant quantity

$$L[U] = \mathrm{tr}\left[\prod_{(n,\mu)\in\mathcal{L}} U_\mu(n)\right] \tag{3.26}$$

The case where $\mathcal{L}$ is the shortest non-trivial closed loop will be of particular importance. In this case, the loop is called a *plaquette* and it is written as

$$U_{\mu\nu}(n) = U_\mu(n)\, U_\nu(n+\hat{\mu})\, U_\mu^\dagger(n+\hat{\nu})\, U_\nu^\dagger(n). \tag{3.27}$$

Plaquettes transform locally as

$$U_{\mu\nu} \;\to\; \Omega(n)\, U_{\mu\nu}\, \Omega^\dagger(n). \tag{3.28}$$

A visual representation of a plaquette is shown in Figure 3.2.

3.4.3 Wilson Gauge Action

Using plaquettes, Kenneth G. Wilson found a discretised version of the gauge action (3.6). In his 1974 paper [20] he introduced the *Wilson gauge action* as a sum over all plaquettes, where each plaquette is counted in only one orientation:

$$S_{\mathrm{G}}[U] = \frac{2}{g^2}\sum_{n\in\Lambda}\sum_{\mu<\nu}\mathrm{Re}\,\mathrm{tr}\left[\mathbb{1} - U_{\mu\nu}(n)\right]. \tag{3.29}$$

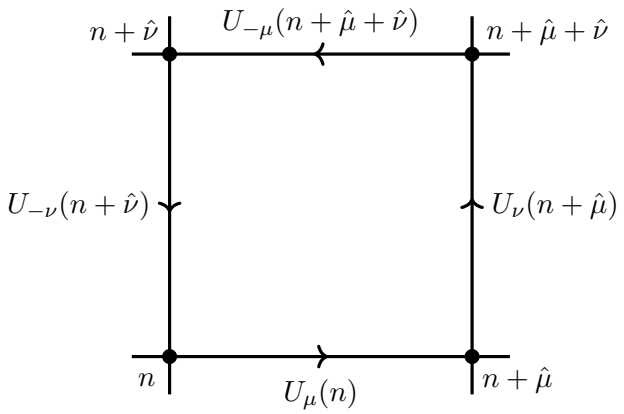


Figure 3.2: A plaquette consists of four links that form a closed loop.

As this expression is gauge invariant, lattice computations using it do not require gauge fixing.

To see that (3.29) reduces to the continuum gauge action for $a \to 0$, insert (3.22) into (3.27), apply the Baker-Campbell-Hausdorff (BCH) formula[1], and Taylor expand terms that involve gauge fields as $A_\nu(n+\hat{\mu}) = A_\nu(n) + a\,\partial_\mu A_\nu(n) + \mathcal{O}(a^2)$:

$$\begin{aligned}
U_{\mu\nu} =\ & \exp[iaA_\mu(n)]\exp[iaA_\nu(n+\hat{\mu})]\exp[-iaA_\mu(n+\hat{\nu})]\exp[-iaA_\nu(n)] \\
\overset{\text{BCH}}{=}\ & \exp\Big[iaA_\mu(n) + iaA_\nu(n+\hat{\mu}) - iaA_\mu(n+\hat{\nu}) - iaA_\nu(n) \\
& - \frac{a^2}{2}\left[A_\mu(n), A_\nu(n+\hat{\mu})\right] - \frac{a^2}{2}\left[A_\mu(n+\hat{\nu}), A_\nu(n)\right] \\
& + \frac{a^2}{2}\left[A_\nu(n+\hat{\mu}), A_\mu(n+\hat{\nu})\right] + \frac{a^2}{2}\left[A_\mu(n), A_\nu(n)\right] \\
& + \frac{a^2}{2}\left[A_\mu(n), A_\mu(n+\hat{\nu})\right] + \frac{a^2}{2}\left[A_\nu(n+\hat{\mu}), A_\nu(n)\right] + \mathcal{O}(a^3)\Big] \\
\overset{\text{Taylor}}{=}\ & \exp\Big[ia^2\,\partial_\mu A_\nu(n) - ia^2\,\partial_\nu A_\mu(n) + a^2\left[A_\mu(n), A_\nu(n)\right] + \mathcal{O}(a^3)\Big] \\
=\ & \exp\Big[\,ia^2 F_{\mu\nu}(n) + \mathcal{O}(a^3)\Big] \\
=\ & \mathbb{1} + ia^2 F_{\mu\nu}(n) - a^4 F_{\mu\nu}^2(n) + \mathcal{O}(a^5)
\end{aligned} \tag{3.30}$$

Inserting this into the Wilson action yields

$$\begin{aligned}
S_{\text{G}}[U] =\ & \frac{2}{g^2}\sum_{n\in\Lambda}\sum_{\mu<\nu}\text{Re}\,\text{tr}\left[a^4 F_{\mu\nu}^2(n) + \mathcal{O}(a^5)\right] \\
=\ & \frac{a^4}{2g^2}\sum_{n\in\Lambda}\sum_{\mu,\nu}\text{tr}\left[F_{\mu\nu}^2(n)\right] + \mathcal{O}(a^2) \\
\xrightarrow{a\to 0}\ & \frac{1}{2g^2}\int \text{d}^4x\ \text{tr}\left[F_{\mu\nu}^2(x)\right] \ =\ S_{\text{G}}[A],
\end{aligned} \tag{3.31}$$

as desired.

[1] $\exp(X)\exp(Y) = \exp\big(X + Y + \frac{1}{2}[X,Y] + ...\big)$ where "..." abbreviates terms of second order in X and Y.

This first discretisation of the gauge action was a vital step towards a lattice gauge theory of QCD but is an approximation only to order $\mathcal{O}(a^2)$. An improvement that includes also higher order terms is given by the Symanzik action [21], which is not used in this work.

3.4.4 The Wilson loop

The operator that allows for the computation of the static quark-antiquark potential is the Wilson loop. It is a product of link variables along a closed path on the lattice. We decompose it into two spatial Wilson lines $S^\dagger$ and S and two temporal transporters $T^\dagger$ and T. The latter can be shown to be the heavy quark propagator in the limit of infinitely heavy quark masses. As the names suggest, the temporal transporters consist exclusively of links in the temporal direction, while each spatial Wilson line connects two points in space, $\mathbf{x}$ and $\mathbf{y}$, along a path $\mathcal{C}_{\mathbf{x},\mathbf{y}}$ that lies in a $t = \text{const.}$ hyperplane. In terms of link variables, the spatial and temporal lines can be written as

$$S(\mathbf{x},\mathbf{y},t) \;=\; \prod_{(\mathbf{k},i)\,\in\,\mathcal{C}_{\mathbf{x},\mathbf{y}}} U_i(\mathbf{k},t), \tag{3.32}$$

$$T(\mathbf{x},t_1,t_2) \;=\; \prod_{j=t_1}^{t_2-1} U_4(\mathbf{x},j). \tag{3.33}$$

The (traced) Wilson loop consisting of these lines is given by

$$W_{\mathcal{L},r,T} \;\equiv\; \mathrm{tr}\left[S(\mathbf{x},\mathbf{y},t)\,T(\mathbf{y},t,t+T)\,S^\dagger(\mathbf{x},\mathbf{y},t+T)\,T^\dagger(\mathbf{x},t,t+T)\right] \tag{3.34}$$

where $r = |\mathbf{x}-\mathbf{y}|$. This is a specific realisation of (3.26) and therefore gauge invariant.

Physically, $S(\mathbf{x},\mathbf{y},t)$ corresponds to the creation of a quark-antiquark pair at locations $\mathbf{x}$ and $\mathbf{y}$ at time t, and $S^\dagger(\mathbf{x},\mathbf{y},t+T)$ corresponds to the annihilation of these particles at time $t+T$. However, this correspondence is not exact, as the Λ_η^ϵ quantum numbers of the state created by S depend on the exact shape of S. In general, the state created by S receives contributions also from excited states. Nevertheless $S(\mathbf{x},\mathbf{y},t)$ does have the gauge property of the continuum creation operator $\psi(\mathbf{x},t)\bar{\psi}(\mathbf{y},t)$:

$$S(\mathbf{x},\mathbf{y},t) \;=\; \prod_{(\mathbf{k},i)\,\in\,\mathcal{C}_{\mathbf{x},\mathbf{y}}} U_i(\mathbf{k},t) \;\to\; \Omega(\mathbf{x},t)\,S(\mathbf{x},\mathbf{y},t)\,\Omega^\dagger(\mathbf{y},t) \tag{3.35}$$

$$\psi(\mathbf{x},t)\,\bar{\psi}(\mathbf{y},t) \;\to\; \Omega(\mathbf{x},t)\,\psi(\mathbf{x},t)\,\bar{\psi}(\mathbf{y},t)\,\Omega^\dagger(\mathbf{y},t) \tag{3.36}$$

Using temporal gauge, where $U_4(n) = \mathbb{1}\;\forall n \in \Lambda$, we see that the Wilson loop is the correlator of its spatial lines:

$$\langle W_{\mathcal{L},r,T}\rangle \;=\; \langle\Omega|\,\mathrm{tr}\left[S^\dagger(\mathbf{x},\mathbf{y},t+T)S(\mathbf{x},\mathbf{y},t)\right]|\Omega\rangle \tag{3.37}$$

$$= \sum_n \langle\Omega|\; e^{TH} S^\dagger_{ab}(\mathbf{x},\mathbf{y},t)\, e^{-TH}\; |n\rangle\langle n|\, S_{ba}(\mathbf{x},\mathbf{y},t)\,|\Omega\rangle \tag{3.38}$$

$$= \sum_n \langle\Omega|\, S^\dagger_{ab}(\mathbf{x},\mathbf{y},t)\;|n\rangle\langle n|\, S_{ba}(\mathbf{x},\mathbf{y},t)\,|\Omega\rangle\; e^{-TE_n} \tag{3.39}$$

$$\equiv \sum_n c_n\, e^{-TE_n} \tag{3.40}$$

where H acts as a Hamiltonian of the system, and $\{|n\rangle\}$ is a complete set of eigenstates of H that have non-vanishing overlap with S.

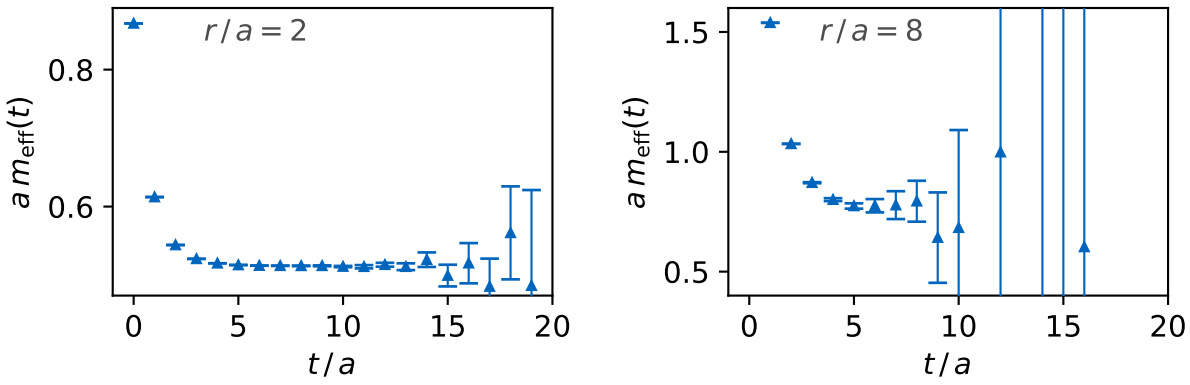


Figure 3.3: Effective mass computed from planar Wilson loops as a function of time. For large values of time t and a fixed quark-antiquark separation r, the effective mass approaches a constant value, corresponding to the ground state energy of the system. However, the uncertainties become very large at large values of t, rendering the determination of the ground state energy difficult. This is true in particular for large values of r, where the uncertainties become large already at smaller values of t (Compare the situation for $r = 2$, as shown in the left panel, to the one for $r = 8$, shown in the right panel).

3.5 Computation of Static Potentials

Knowing interpolators O and $\bar{O}$ that correspond to the Hilbert space operators O and $O^\dagger$ annihilating and creating a hadron state, one can find the ground state energy of that hadron by studying the correlator

$$C(t) \equiv \langle O(t)\, \bar{O}(0) \rangle = \sum_n \langle \Omega | O | n \rangle \langle n | O^\dagger | \Omega \rangle \; e^{-tE_n} \tag{3.41}$$

and the effective mass

$$m_{\text{eff}}(t) \equiv \ln \frac{C(t)}{C(t+1)} \tag{3.42}$$

in the large-t limit.

In the case of a static quark-antiquark pair located at positions $\mathbf{x}$ and $\mathbf{y}$, the interpolator $O(t)$ is given by a spatial Wilson line running from $\mathbf{x}$ to $\mathbf{y}$ at time t. The correlator $C(t)$ than is a Wilson loop:

$$C(t) \;=\; \langle W_{\mathcal{L},r,t} \rangle \;=\; \sum_{n=0} c_n\, e^{-E_n t} \;=\; e^{-E_0 t} \left(c_0 + \sum_{n \geq 1} c_n\, e^{-\Delta E_n t} \right) \tag{3.43}$$

As expected from Section 3.3, for $t \to \infty$ the correlator receives only contributions from the energetically lowest state that has non-vanishing overlap with the state created by S. This leads to the effective mass (3.42) reducing to this state's energy for large times t:

$$\lim_{t\to\infty} m_{\text{eff}} = \lim_{t\to\infty} \ln \frac{\langle W_{\mathcal{L},r,t}\rangle}{\langle W_{\mathcal{L},r,t+a}\rangle} \tag{3.44}$$

$$= \lim_{t\to\infty} \ln \left[e^{aE_0} \frac{1+\sum \frac{c_n}{c_0} e^{-\Delta E_n t}}{1+\sum \frac{c_n}{c_0} e^{-\Delta E_n (t+a)}} \right] \tag{3.45}$$

$$= aE_0 + \lim_{t\to\infty} \ln \left[\frac{1+\sum \frac{c_n}{c_0} e^{-\Delta E_n t}}{1+\sum \frac{c_n}{c_0} e^{-\Delta E_n (t+a)}} \right] \tag{3.46}$$

$$= aE_0 \tag{3.47}$$

where $\Delta E_n \equiv E_n - E_0$. Repeating this computation at different values of r yields the static potential $aV(r) = aE_0(r)$.

The above derivation makes no assumptions regarding the shape of the path S. However, depending on this shape, the resulting state has a smaller or larger overlap with the ground state. If one uses spatial lines that correspond to different Λ_η^ϵ quantum numbers, the large-t limit will yield the energy E_0 of the energetically lowest state with which the corresponding Wilson loop has non-vanishing overlap. Therefore, Wilson loops can be used to compute the static potentials of the ground state as well as gluonic excitations of quarkonium.

If one takes the Wilson loop to be planar, the corresponding state has a non-vanishing overlap with the ground state. In Figure 3.3 we show the effective mass obtained from planar Wilson loops for our set of 6000 lattice gauge configurations[2]. As expected, the effective mass falls off with increasing t, reaching a plateau at large values of t. In principle, the ground state energy can be read off in the plateau region. However, the uncertainties in effective mass become very large in this region, rendering the identification of the ground state energy difficult. This effect is particularly pronounced at larger values of r, as is reflected in the effective mass for $r = 8$ (right panel of Figure 3.3) in comparison to the one for $r = 2$ (left panel of Figure 3.3).

This can be mitigated by choosing spatial lines with large ground state overlap. Than the effective mass is at its plateau value already at small times t, where the uncertainties are smallest. Several realisations of this idea have already shown first successes [4, 5, 22].

Despite the large uncertainties in the case of straight spatial lines, the static potential, which is shown in Figure 3.4, is well described by a Cornell potential fit. As a consequence of the uncertainties in the effective mass, the uncertainties in the static potential tend to become larger at larger values of r. Within these uncertainties, our data agrees with that of Schlosser and Wagner [5], who used optimised Wilson loops for their computation (cf. lower panel of Figure 3.4). We will use the data we obtained from planar Wilson loops as a benchmark for the results of our neural network-based computation.

3.5.1 Off-axis Separations

The data shown in Figure 3.4 is calculated for a quark and antiquark that are displaced by a distance $r \in \mathbb{Z}^+$ along a common spatial axis. In such a case one speaks of an *on-axis* separation, otherwise of an *off-axis* separation. It is desirable to treat off-axis separations as this allows for finer spatial resolution. Our neural network approach is suitable for the treatment of both on-axis and off-axis separations.

[2] Details on these lattice gauge configurations are provided in Section 3.9.

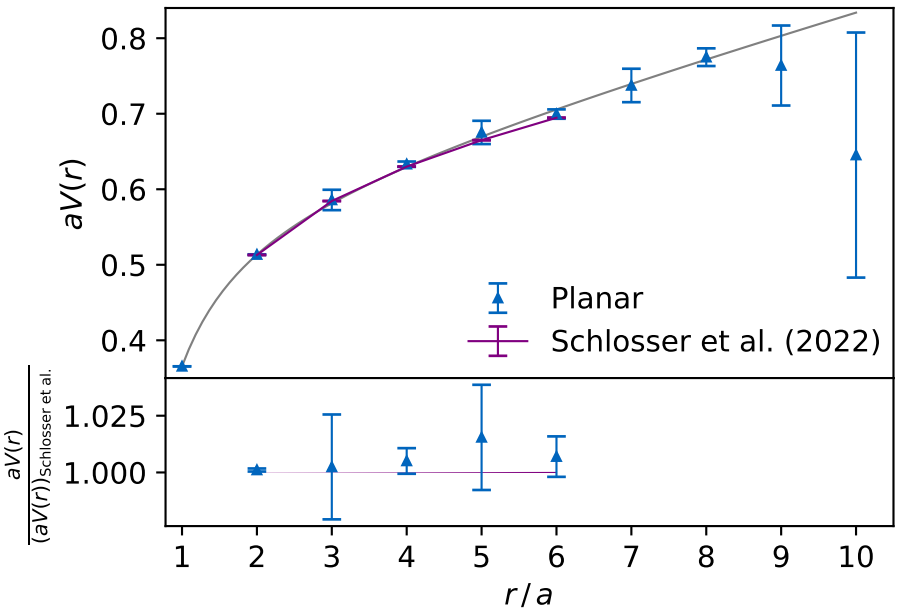


Figure 3.4: Upper panel: The static potential obtained from the large-t values of the effective mass at the different values of r, using planar Wilson loops. The grey curve is the best-fit Cornell potential for these data points, and the purple curve is the data obtained by Schlosser and Wagner [5]. **Lower panel**: The data from planar Wilson loops normalised by the data obtained by [5]. The purple stripe indicates the 1σ-interval from [5].

3.6 Scale Setting

In lattice QCD, values of observables a priory dimensionless. In order to determine their physical values one has to perform some form of scale setting. This can be done, for example, by identifying some observables with their experimental values. Suitable observables are, e.g. well-known hadron masses.

Another way of scale setting makes use of the static quark-antiquark potential. The force corresponding to the Cornell potential (2.32) is given by

$$F(r) = -\frac{\mathrm{d}V(r)}{\mathrm{d}r} = -\frac{A}{r^2} - \sigma. \tag{3.48}$$

This is used to define the so-called Sommer scale r_0 according to [23] through

$$-r_0^2\, F(r_0) \;=\; 1.65. \tag{3.49}$$

Using the values of A and $\sigma\, a^2$, which are known from the Cornell potential fit, definition (3.49) implies

$$A + \sigma\, r_0^2 \;=\; 1.65 \tag{3.50}$$

$$\Rightarrow \frac{r_0}{a} \;=\; \sqrt{\frac{1.65 - A}{\sigma a^2}}. \tag{3.51}$$

The right-hand side is known from the lattice data, and the value of the Sommer scale is $r_0 \approx 0.5\,\mathrm{fm}$. Therefore the lattice spacing a can be determined from (3.51).

3.7 Monte Carlo Integration

Now that we have introduced the observables that are relevant to the Wilson loop calculation, we discuss how to measure these, i.e. how to obtain numerical values for the observables. In particular, we look for a way to numerically approximate the Euclidean correlator

$$\langle O \rangle \;=\; \frac{\int \mathcal{D}U\, O[U]\, e^{-S[U]}}{\int \mathcal{D}U\, e^{-S[U]}}. \tag{3.52}$$

This is commonly achieved through Monte Carlo integration, which approximates a definite Riemann integral

$$I \;=\; \frac{1}{b-a}\int_a^b f(x)\,\mathrm{d}x \qquad (a, b \in \mathbb{R}) \tag{3.53}$$

by a sum

$$\frac{1}{N}\sum_{i=0}^{N} f(x_i) \approx I \tag{3.54}$$

where the $x_i \in [a, b]$ are sampled from a uniform probability distribution. More generally, one can apply importance sampling, where the x_i are sampled from a probability distribution $w : [a, b] \to [0, 1]$. The integrand than has to be divided by $w(x_i)$:

$$I \;=\; \frac{1}{b-a}\int_a^b \frac{f(x)}{w(x)}\, w(x)\,\mathrm{d}x \;\approx\; \frac{1}{N}\sum_{P(x_i)\,\propto\, w(x_i)} \frac{f(x_i)}{w(x_i)} \tag{3.55}$$

This allows us to approximate the path integral (3.52) by a sum over field configurations. The Boltzmann factor $\exp(-S_e)$ is taken into account by sampling the field configurations from the corresponding Boltzmann distribution. We can therefore approximate the Euclidean correlator (3.52) by

$$\langle O \rangle \;\approx\; \frac{1}{N}\sum_{P(U_n)\,\propto\,\exp(-S[U_n])} O[U_n]. \tag{3.56}$$

This means that we can calculate the Euclidean correlator as an average over suitably sampled gauge field configurations.

3.7.1 Sampling a Set of Gauge Field Configurations

In order to find an ensemble of gauge field configurations $\{\, U_n \,\}$ that follows the Boltzmann distribution, one can start at an arbitrary field configuration U_0 and construct a sequence of field configurations U_n in order of increasing n. In this process one typically imposes a transition probability

$$P(\, U_{n+1} = U' \,|\, U_n = U \,) \;=\; T(\, U' \,|\, U \,) \tag{3.57}$$

that only depends on the current field configuration $U_n = U$ and the potential next field configuration U'. Equation (3.57) is the so-called *Markov property*, and a process that satisfies it is called a *Markov process*. The generation of a new U_{n+1} from the previous U_n is referred to as a *Monte Carlo step*. As field configurations that are only few Monte Carlo steps apart are expected to be correlated, only every kth configuration that is generated is added to the ensemble that is used for the Monte Carlo integration.

The $T(\,U'\,|\,U\,)$ have to satisfy the positivity and normalisation properties characteristic of probabilities, i.e.

$$0 \;\leq T(\,U'\,|\,U\,) \;\leq 1, \qquad \sum_{U'} T(\,U'\,|\,U\,) \;=\; 1. \tag{3.58}$$

In equilibrium, the probability to enter a configuration U' at a given step n is equal to the probability of leaving U':

$$\sum_{U} T(\,U'\,|\,U\,)\,P(U) \;=\; \sum_{U} T(\,U\,|\,U'\,)\,P(U') \;=\; P(U'). \tag{3.59}$$

This shows that the equilibrium distribution $P(U)$ is fixed point of the Markov process, i.e. further Monte Carlo steps map the distribution to itself. A sufficient condition for equilibrium is the detailed balance condition, which requires equality of the individual terms in (3.59):

$$T(\,U'\,|\,U\,)\,P(U) \;=\; T(\,U\,|\,U'\,)\,P(U'). \tag{3.60}$$

A Markov chain Monte Carlo method frequently used to sample field configurations that follow the Boltzmann distribution is the *Metropolis algorithm*. First, a candidate configuration U' is selected based on an a priori selection probability $T_0(U'|U_n)$. This candidate is then either accepted as the new field configuration U_{n+1}, or rejected, in which case one sets $U_{n+1} = U_n$. The acceptance probability is given by

$$T_{\mathrm{A}}(U'|U_n) = \min\left(\left\{1\,,\,\frac{T_0(U_n|U')\exp(-S[U'])}{T_0(U'|U_n)\exp(-S[U_n])}\right\}\right). \tag{3.61}$$

In the case of lattice QCD, this can be realised as follows: starting from the current configuration U_n, one considers a candidate configuration $U' \in \mathrm{SU}(3)$ that differs from U_n only slightly. This can be achieved by generating U' as $U'_\mu(n) = XU_\mu(n)$ where $X \in \mathrm{SU}(3)$ lies in the vicinity of the identity. One then calculates a random number r from a uniform distribution on $[0, 1)$ as well as the difference ΔS of the actions associated with U_n and U' respectively. One accepts U' if $r \leq \exp(-\Delta S)$:

$$U_{n+1} = \begin{cases} U', & \text{if } r \leq \exp(-\Delta S) \\ U_n, & \text{otherwise} \end{cases} \tag{3.62}$$

This means that for $\Delta S \leq 0$, i.e. if the action associated with U' is smaller than the one associated with U_n, the candidate U' is always accepted. In the other case, i.e. if $\Delta S > 0$, the candidate may be accepted or rejected. This behaviour may be understood to imitate quantum fluctuations.

3.8 The Continuum Limit

In order to make predictions for continuum QCD from lattice computations, one has to take the continuum limit $a \to 0$. This, of course, means that the lattice gauge theory has to have a well-defined continuum limit and that in this limit, one recovers QCD. One can show that this is indeed the case for lattice QCD [24].

In practice, this means that one has to compute observables at several values of the lattice spacing a and then extrapolate to $a \to 0$, or, in terms of the inverse coupling $\beta = \frac{6}{g^2}$, to $\beta \to \infty$. If one keeps the physical volume constant during this process, one has to use lattices with a larger number of lattice sites when decreasing the value of a.

3.9 Lattice Setup

In this work, we use 6000 quenched gauge field configurations on a $20^3 \cdot 40$ lattice. These configurations were generated by the MILC code [25] using the Wilson action. The inverse coupling for these field configurations is given by $\beta = 6.284$, corresponding to a lattice spacing of $a = 0.060\,\text{fm}$. The scale setting was done through [26], yielding the Sommer scale in lattice units as $r_0/a = 8.306$. Other scale setting methods yield e.g. $r_{0,\text{Necco,Sommer}}/a = 8.326$ [27] and $r_{0,\text{Kaczmarek}}/a = 8.320$ [28].

4 The Neural Network

In Chapter 3, we saw that information on the ground state of the static quark-antiquark pair is contained in the planar Wilson loop, but that a Wilson loop with larger ground state overlap is desirable to extract the static potential. Our objective now is to optimise the shape of the Wilson loop towards maximum ground state overlap. Such an optimisation task can be completed by a neural network. In this chapter, we briefly introduce some nomenclature regarding neural networks, before describing in detail a neural network suitable for the Wilson loop optimisation.

4.1 Introduction to Neural Networks

Artificial neural networks consist of so-called neurons, which are typically arranged into layers. Each neuron corresponds to a real number called its activation. The activations in the first layer are inputs to the code. Each layer is a function that maps the activations of the preceding layer to new activations. The activations of the final layer, which is the output layer, correspond to the likelihoods the code attributes to the possible outcomes. Thus the neural network is a composite function that maps the activations of the input layer to activations of the output layer.

Let $a_j^{(l)} \in \mathbb{R}$ denote the activation of the jth neuron in the lth layer. Than there typically[1] exist a complex matrix $W^{(l)}$ and a complex vector $\mathbf{b}^{(l)}$ such that

$$\mathbf{a}^{(l)} = f^{(l)}\left(W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{4.1}$$

i.e. the activation $a_j^{(l)}$ is computed from a weighted sum of the activations of the neurons in the $(l-1)$th layers plus a bias $b_j^{(l)}$. The function $f^{(l)}$ mapping this sum to $a_j^{(l+1)}$ is called the *activation function* at layer l. The weights and biases of each layer are parameters of the neural network.

4.1.1 Training

The process of finding the optimal values of the neural network's parameters p is referred to as training. For the training, one defines a *loss function* L that maps the parameters to a real number called the cost $L(p, \mathbf{a}^{(l)})$. The objective of the training process is to minimise the loss function, which should, therefore, have a minimum at the parameters' optimal values. The training dataset is split into several batches, which one uses in the following steps of a backpropagation algorithm:

1. **Forward pass**: The data of the current batch is passed through the neural network. The output of the neural network is then used to compute the associated cost.
2. **Backwards pass**: Propagating layer by layer backwards through the network, one computes the gradient of the loss function with respect to the parameters of the neural network.
3. **Weight update**: The parameters are updated e.g. using the method of gradient descent.
4. **Repetion** of these steps for the next batch.

One complete pass through all batches is called an epoch.

[1] Our neural network contains a bilinear layer, which is not of this form. The considerations presented here apply to our neural network nonetheless.

There are several so-called hyperparameters whose values are fixed for the entire training process, and which influence the speed and effectiveness of the training. We briefly discuss the impact of each hyperparameter relevant to us, and state the respective values we used:

- **Batch size**: The batch size is the number of data samples in each batch. It determines the amount of data that is taken into account for each weight update. Larger batch sizes dampen the noise in the loss function, but entail a larger memory usage and decrease the number of training steps within an epoch. In our case, one batch contains one complete lattice gauge configuration. We found that although considering more data before each weight update improves the effectiveness of the training, the GPU resources available do not admit batches of several gauge configurations. This is why we define *superbatches*.

- **Superbatch size**: The superbatch size is the number of batches that we group together to what we call a superbatch. Instead of updating the weights after every batch, we store the gradient computed in the backwards pass for each batch of a superbatch, and use the gradients computed within a superbatch to update the weights once per superbatch. We apply superbatching only when training models with a large number of parameters, and choose the superbatch size for each such training individually.

- **Learning rate**: The learning rate determines how much the weights are changed in the weight update. We perform the weight update with the Adam optimizer [29] at a learning rate of 10^{-1}.

- **Number of epochs**: While it requires a certain number of epochs for the parameters of the neural network to converge at their optimal values, a very large number of epochs can lead to overfitting, i.e. to the neural network learning the specific data at hand instead the underlying patterns. This is, however, not a major concern in our case.

4.2 Modification of the Planar Wilson Loop

We wish to modify the planar Wilson loop

$$W_{\mathcal{L},r,T} \;=\; \mathrm{tr}\left[S(\mathbf{x},\mathbf{y},t)\,T(\mathbf{y},t,t+T)\,S^{\dagger}(\mathbf{x},\mathbf{y},t+T)\,T^{\dagger}(\mathbf{x},t,t+T)\right]$$

such that the overlap with the ground state is maximised.

The temporal transporters $T(\mathbf{y},t,t+T)$ and $T^{\dagger}(\mathbf{x},t,t+T)$ are present in the Wilson loop because they are the infinite mass limit of the heavy quark propagators. Modifying them would mean that we are no longer describing a quark-antiquark pair. Therefore, we leave them unchanged and modify only the shape of spatial Wilson lines.

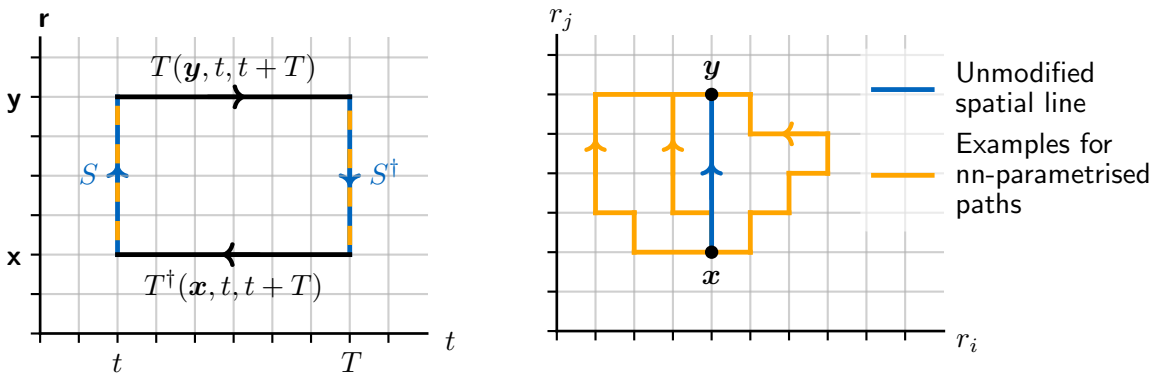
As shown in Section 3.4.4, the spatial Wilson lines obey the same gauge transformation law as $\psi(\mathbf{x},t)\bar{\psi}(\mathbf{y},t)$, as it has to be if we describe a quark-antiquark pair at spatial positions $\mathbf{x}$ and $\mathbf{y}$. Moreover, this gauge property ensures that the Wilson loop is gauge invariant. For the same reasons, the modified spatial segments S of the Wilson loop must have this gauge property, i.e.

$$S(\mathbf{x},\mathbf{y},t) \;\to\; \Omega(\mathbf{x},t)\,S(\mathbf{x},\mathbf{y},t)\,\Omega^{\dagger}(\mathbf{y},t) \qquad (i=1,2). \tag{4.2}$$

Since any superposition of paths that start at $\mathbf{x}$ and end at $\mathbf{y}$ possesses gauge property (4.2), we look the a superposition that maximises the overlap with the ground state. As neural networks are designed for optimisation problems, we parametrise the Wilson loops by a neural network and find the optimal values of the parameters by training the latter. This idea is illustrated in Figure 4.1.

At first, we will assume that $\mathbf{x}$ and $\mathbf{y}$ are separated on-axis. We will generalise to off-axis separations in Section 4.6.

For the realisation of this neural network, we use the PyTorch machine learning framework [30].



(a) **(b)**

Figure 4.1: Visualisation of how the neural network (orange lines) replaces the straight Wilson lines (blue lines) with a superposition of differently shaped spatial lines (orange lines). **(a)** The neural network replaces the straight spatial Wilson lines with superpositions of spatial paths that lie in the same equal-time hyperplanes as the straight Wilson lines. The temporal transporters (black lines) remain unchanged. **(b)** In the r_i-r_j plane (where $i \neq j$), the neural network parametrised paths deviate from the unmodified path.

4.3 Input Data

The input to our neural network consists of three elements. The first two of these are:

- a set of link variables $\{U_\mu(x)\}$ that describes the gauge field configuration,
- the straight Wilson line $\phi^{(0)} \equiv S(\mathbf{x}, \mathbf{y})$.

The nth layer of the neural network maps a set of superpositions $\{\, \phi_i^{(n)} \mid 1 \leq i \leq N_{\text{in}} \,\}$ to a set of superpositions $\{\, \phi_j^{(n+1)} \mid 1 \leq j \leq N_{\text{out}} \,\}$ where N_{in} and N_{out} denote the numbers of so-called *channels* for the input and output of the layer respectively. Each $\phi_i^{(n)}$ will be a superposition of paths from $\mathbf{x}$ to $\mathbf{y}$.

For the initial layer a set $\{\phi_i^{(0)}\}$ needs to be provided. This is the third element of the input:

- A set of Wilson lines, each with one plaquette insertion, $\{\phi_i^{(0)}\}$, where i iterates over all possible plaquette insertions. We do not include plaquettes that contain links in the temporal direction.

In a straight Wilson line of length r, there are $r+1$ possible positions to insert a plaquette. Being careful to not count any plaquette twice, there are $24 + 16\,r$ possible plaquette insertions, i.e. the number of channels in the input of the first layer is $24 + 16\,r$. As this number as well as the possible paths that can be constructed between $\mathbf{x}$ and $\mathbf{y}$ depends on r, we train one neural network for each value of r.

We train and evaluate the neural network on the same dataset where each batch consists of one lattice gauge configuration.

4.4 Layers

The neural network is supposed to find the superposition of differently shaped paths that maximises the ground state overlap. In order to ensure that paths of various different shapes can be included

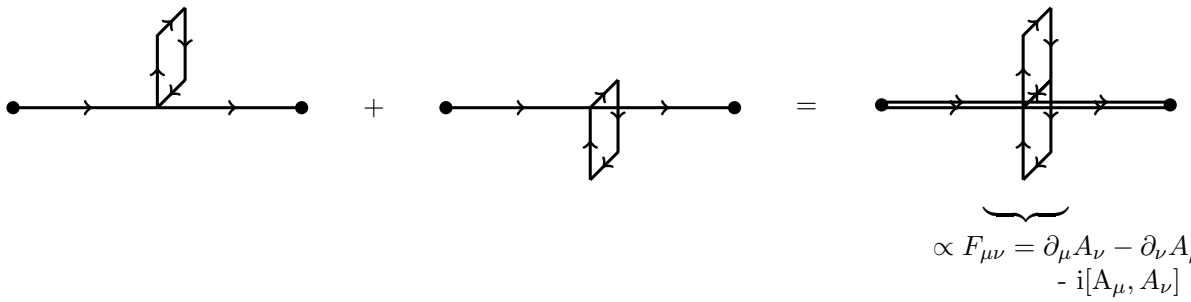


Figure 4.2: From the input data, which contains straight spatial Wilson lines with one plaquette insertion each, a linear layer may produce results that are proportional to field insertions.

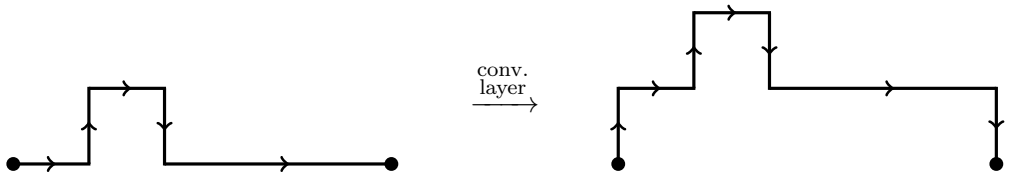


Figure 4.3: A convolutional layer may produce paths that extend further from the quark-antiquark separation axis.

in this superposition, we use several types of layers. A similar approach by Favoni et al. [31] has already led to promising results.

Each layer of our neural network individually preserves gauge property (4.2):

$$\phi_i^{(n)} \;\to\; \Omega(x)\, \phi_i^{(n)}\, \Omega^\dagger(y) \tag{4.3}$$

This will be verified explicitly for each of the layer types described below.

Furthermore, we wish to ensure that the ground state is included in the final superposition of paths. Since we know from Section 3.3 that the ground state is contained in the straight Wilson line $\phi^{(0)}$, we add a bias term proportional to $\phi^{(0)}$ to each layer of the network.

In the gauge where $\phi^{(0)} = \mathbb{1}$ the layers described below reduce to layers commonly used in the field of machine learning.

Linear Layer

A linear layer changes the $\phi_i^{(n)}$ according to

$$\phi_i^{(n+1)} = \sum_j \omega_{ij}^{(n)} \phi_j^{(n)} \;-\; b_i^{(n)} \phi^{(0)} \tag{4.4}$$

where $\omega_{ij}^{(n)}, b_i^{(n)} \in \mathbb{C}$ are the weights and biases of the layer. Because of translational and rotational invariance, we impose the same weights for all $\mathbf{x}$ and $\mathbf{y}$ at a fixed distance $r = |\mathbf{x} - \mathbf{y}|$.

Assuming that the input to the layer, i.e. $\phi^{(0)}$ and the $\phi_j^{(n)}$, transform under gauge transformations as required, the output of a linear layer transforms as desired:

$$\phi_i^{(n+1)} \;\to\; \Omega(x)\, \phi_i^{(n+1)}\, \Omega^\dagger(y) \tag{4.5}$$

As sketched in Figure 4.2, a linear layer may create objects proportional to field insertions $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - i[A_\mu, A_\nu]$, which can create or remove excited states.

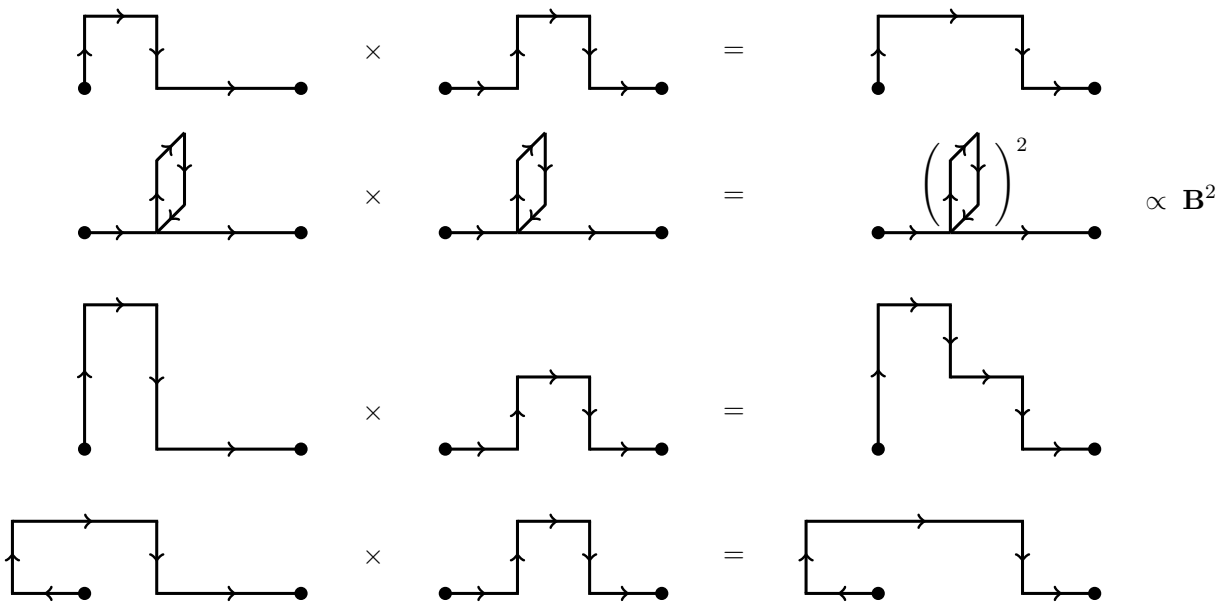


Figure 4.4: Example sketches of how a bilinear layer may "combine" paths.

Convolutional Layer

We define a convolutional layer by

$$\phi_i^{(n+1)} = \sum_{\substack{j \\ \mu=0,\,\pm x,\,\pm y,\pm z}} \omega_{ij\mu}^{(n)}\, U_\mu(x)\, \phi_j^{(n)}(x+\hat{\mu}, y+\hat{\mu})\, U_\mu^\dagger(y) - b_i^{(n)}\phi^{(0)} \tag{4.6}$$

where $\omega_{ij\mu}^{(n)}, b_i^{(n)} \in \mathbb{R}$ are the weights and biases of the layer. For the $\mu = 0$ term we define $U_0(x) \equiv \mathbb{1}$. If this layer is preceded by a linear layer, the presence of the $\mu = 0$ term ensures that the paths found by the preceding layer may appear also in the output of the convolutional layer.

The convolutional layer preserves the gauge property (4.3):

$$\begin{aligned}\phi_i^{(n+1)} \to \sum_{j,\mu}\Big[& \omega_{ij\mu}^{(n)}\, \Omega(x)\, U_\mu(x)\, \Omega^\dagger(x+\hat{\mu})\, \Omega(x+\hat{\mu})\, \phi_j^{(n)}(x+\hat{\mu}, y+\hat{\mu})\, \Omega^\dagger(y+\hat{\mu}) \\ & \Omega(y+\hat{\mu})\, U_\mu^\dagger(y)\, \Omega^\dagger(y) \;-\; b_i^{(n)}\, \Omega(x)\, \phi^{(0)}\, \Omega^\dagger(y)\Big] \\ = \;& \Omega(x)\, \phi_i^{(n+1)}\, \Omega^\dagger(y)\end{aligned} \tag{4.7}$$

As illustrated in Figure 4.3, convolutional layers allow for paths that extend further from the straight Wilson line.

Bilinear Layer

We define a bilinear layer by

$$\phi_i^{(n+1)} = \sum_{j,k} \omega_{ijk}^{(n)} \phi_j^{(n)} \phi^{(0)\dagger} \phi_k^{(n)} \;-\; b_i^{(n)}\phi^{(0)} \tag{4.8}$$

Model name	Terms included in loss function	Layer setup	# Channels
NN 1	"tower of exponentials"	1× Linear, 1× Convolutional	$x \to 5$ $5 \to 1$
NN 2	"tower of exponentials" & "flat effective mass"	1× Linear, 1× Convolutional	$x \to 5$ $5 \to 1$
NN 3	"tower of exponentials"	1× Linear, 2× Convolutional	$x \to 5$ $5 \to 5 \to 1$
NN 4	"tower of exponentials" & "flat effective mass"	1× Linear, 2× Convolutional	$x \to 5$ $5 \to 5 \to 1$

Table 4.1: Models used for the computation of the Σ_g^+ potential.

where $\omega_{ijk}^{(n)}, b_i^{(n)} \in \mathbb{R}$ are the weights and biases of the layer. Again, we demonstrate that this layer transforms according to (4.3) under gauge transformations:

$$\phi_i^{(n+1)} \to \sum_{j,k} \omega_{ijk}^{(n)}\, \Omega(x)\, \phi_j^{(n)}\, \Omega^\dagger(y)\, \Omega(y)\, \phi^{(0)\dagger}\, \Omega^\dagger(x)\, \Omega(x)\, \phi_k^{(n)} \;-\; b_i^{(n)}\, \Omega(x)\, \phi^{(0)}\, \Omega^\dagger(y) \tag{4.9}$$

$$= \Omega(x)\, \phi_i^{(n+1)}\, \Omega^\dagger(y) \tag{4.10}$$

Some paths that can be created by bilinear layers are sketched in Figure 4.4. These paths can be complex in shape, and thereby diversify the set of paths that the neural network can construct.

Initial Values of the Parameters

In order to consider the contributions of different paths already at the beginning of the training process, we initialise the weights and biases according to a Gaussian distribution with standard deviation 0.3 and means 0 and -1, respectively. The values 0 and -1 are the values for which the neural network simply returns a straight Wilson line.

Layer setup

We test and compare multiple layer setups. The first layer is always a linear layer that reduces the $24(r+1)$ input channels to 5-10 output channels. This reduces the RAM and computing time needed by the following layers. The models used for the ground state computation are listed in Table 4.1. For the ground state computation, we used only linear and convolutional layers, but multilinear layers are relevant to our treatment of hybrid states described in Chapter 6.

In Appendix A we provide explicit code to illustrate a possible implementation of the neural network and its layers.

4.5 The Loss Function

The loss function should be chosen such that the neural network seeks the maximum overlap with the energetically lowest state. In addition, the loss function can be designed such that the noise associated with it is small, in order to allow for effective training. Keeping these intentions in mind, we now discuss the individual terms of the loss function.

Normalised tower of exponentials

As discussed in Section 3.4.4, the correlator as a function of time is expected to take the form $C(t) = \sum_n c_n e^{-E_n t}$. We make use of this by defining a function f by

$$f(t) \equiv \frac{\sum_n c_n e^{-E_n t}}{\sum_n c_n}. \tag{4.11}$$

This function satisfies $f(t) \leq e^{-E_0 t}$, with equality holding if and only if $c_n = 0 \, \forall n \neq 0$ [2]. Hence, maximising it leads to a single exponential, as is the case for the ground state. Moreover, through the normalisation $f(0) = 1$, f is constructed such that it cannot be maximised by maximising the c_n. Multiplied by a negative constant, $f(t)$ therefore yields a suitable candidate for a term in the loss function:

$$-C_1 \sum_t \frac{\langle W_{\mathcal{L},r,t}(p) \rangle}{\langle W_{\mathcal{L},r,0}(p) \rangle} \tag{4.12}$$

where p denotes the parameters of the neural network.

Flat effective mass plot

As discussed in Section 3.5, the effective mass is expected to be constant when the ground state is reached at large times t. As we aim for a Wilson loop that from the beginning contains only the ground state, we demand the sum

$$\sum_t (m_{\text{eff}}(t) - m_{\text{eff}}(t+1))^2 \tag{4.13}$$

to be minimised.

Noise reduction term

In order to reduce the noise in the loss function, we define a noise reduction term

$$\sum_t \frac{\langle \, (W_{\mathcal{L},r,t}(p) - \langle W_{\mathcal{L},r,t}(p) \rangle \,)^2 \rangle}{\langle W_{\mathcal{L},r,t}(p) \rangle^2}. \tag{4.14}$$

Here, $\langle \cdot \rangle$ denotes the mean across a training batch. The effect of this term is to reduce the variance in the traced Wilson loop across a training batch.

Putting the different terms together, the loss function reads

$$L(p) = \underbrace{-C_1 \sum_{t=1}^{20} \frac{\langle W_{\mathcal{L},r,t} \rangle}{\langle W_{\mathcal{L},r,0} \rangle}}_{\text{tower of exponentials}} + \underbrace{C_2 \sum_{t=0}^{5} (m_{\text{eff}}(t) - m_{\text{eff}}(t+1))^2}_{\text{flat effective mass}} + \underbrace{C_3 \sum_{t=0}^{20} \frac{\langle (W_{\mathcal{L},r,t} - \langle W_{\mathcal{L},r,t} \rangle)^2 \rangle}{\langle W_{\mathcal{L},r,t} \rangle^2}}_{\text{noise reduction}} \tag{4.15}$$

where p denotes the set of parameters of the neural network, and $\langle \cdot \rangle$ denotes the arithmetic mean across the training batch. The coefficients C_i serve to adjust the contributions of the different terms. The summation limits for the "flat effective mass" term are chosen such that this term is evaluated in a region with a sufficiently high signal-to-noise ratio.

[2] This can be seen by rewriting the expression as $\sum_n \tilde{c}_n e^{-E_n t}$ where $\sum_n \tilde{c}_n = 1$. Since $E_0 < E_n \, \forall n > 0$, the expression is maximised for $\tilde{c}_n = \delta_{0n}$.

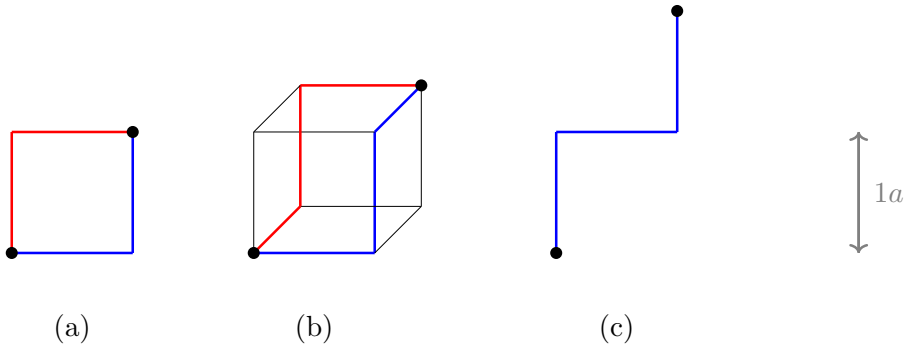


Figure 4.5: (a) and (b): For diagonal off-axis separations, we compute the mean of two paths that can be decomposed into segments like they are depicted here in blue and red, respectively. **(c)**: In order to compute the static potential at separations that are integer multiples of $\sqrt{5}a$, we use single paths that can be decomposed into segments like the one shown here.

4.6 Off-Axis Separations

If the positions $\mathbf{x}$ and $\mathbf{y}$ are off-axis separated, there is no unique shortest spatial path $\phi^{(0)}$ connecting them. Instead, we replace the inputs $\phi^{(0)}$ and $\phi_i^{(0)}$ of the neural network by the means of paths that lie close to the "diagonal" connecting $\mathbf{x}$ and $\mathbf{y}$. We treat the three types of off-axis separations that increase the spatial resolution of the potential the most:

- The distance between $\mathbf{x}$ and $\mathbf{y}$ is an integer multiple of $\sqrt{2}a$, i.e. $\mathbf{x}$ and $\mathbf{y}$ lie on opposite sites of a square on the lattice. In this case, we use the mean of two paths that can be decomposed into segments like that shown in Figure 4.5 (a).
- The distance between $\mathbf{x}$ and $\mathbf{y}$ is an integer multiple of $\sqrt{3}a$, i.e. $\mathbf{x}$ and $\mathbf{y}$ lie on opposite sites of a cube on the lattice. In this case, we use the mean of two paths that can be decomposed into segments like that shown in Figure 4.5 (b).
- The distance between $\mathbf{x}$ and $\mathbf{y}$ is an integer multiple of $\sqrt{5}a$. In this case, we use only one path as input to the neural network, namely the one that consists of segments like the one shown in Figure 4.5 (c).

In a similar fashion, one can compute the static potential at integer multiples of $\sqrt{6}a$, $2\sqrt{2}a$, etc.

4.7 Limitations of the Neural Network Parametrisation

The main limitation that impacted this work was the GPU resources available. During the backwards pass, many intermediate results are stored for the computation of the gradient. This is very RAM demanding, especially if the neural network has many parameters or if the training batches are very large. We could mitigate this problem by introducing superbatches, manually implementing the gradient computation, and using a linear layer as the first layer in the neural network. While this significantly reduces the RAM demands of our code, the GPU RAM required for the training of the neural network remains the dominant limitation to our neural network parametrisation.

5 Results for the Static Quark-Antiquark Potential

In this chapter, we present the results for the quarkonium ground state we obtained from the neural network parametrised Wilson loops described in Chapter 4. We compare the effective masses and static potential computed with neural network parametrised Wilson loops to those computed with planar Wilson loops. Moreover, we investigate the composition of the superposition found by the neural network. We estimate the uncertainties of our results using the jackknifing technique [32].

5.1 Effective Mass and Static Potential

In the logarithmic plot shown in Figure 5.1, the normalised correlators $C(t)/C(0)$ computed with neural network parametrised Wilson loops are closer to a straight line, which would correspond to a single exponential, than those computed with planar Wilson loops. The difference is particularly pronounced at large values of r. This also reflects itself in the effective masses plotted in Figure 5.2. Although not entirely flat, the effective masses corresponding to neural network parametrised Wilson loops are smaller than the ones obtained from planar Wilson loops. This indicates that the neural network successfully increased the ground state overlap.

To compare the different neural network models, consider the right panel of Figure 5.1, which is the same as the central panel, except zoomed in to $t \in [9, 13]$. The layer setup (which is the same for the models NN 1 and NN 2 as well as for NN 3 and NN 4) has a larger impact on the model's effectiveness than the inclusion of the "flat effective mass"-term in the loss function (included for NN 2 and NN 4, but not for NN 1 and NN 3). This suggests that involved neural network structures, which correspond to a greater variety of paths that can appear in the final superposition, are central to the effectiveness of the neural network parameterisation.

The static potentials computed with the neural network models are compared to those computed from planar Wilson loops in Figure 5.3. Most notably, all neural network models lead to much smaller uncertainties in the potential at very large separations $r \geq 8$. The results for separations $r \geq 5$ give the slight suggestion that the potentials obtained with neural networks tend to be smaller than the ones corresponding to planar Wilson loops. This is similar to the effect of smearing on the static potential.

Finally, Figure 5.4 shows the static quark-antiquark potential computed with neural network parametrised Wilson loops for off-axis separations compared to the on-axis data for planar Wilson loops. The on- and off-axis data points are well described by a common Cornell potential fit.

5.2 Composition of the Neural Network Parametrised Superposition

We now investigate the superposition of paths corresponding to the neural network model NN 4.

Firstly, Figure 5.5 shows all paths that appear in the superposition for $r = 3$ and $r = 10$, with the opacities of the paths being proportional to the absolute values of the associated (complex) weights. We see that the overall superposition is symmetric about the separation axis and that paths that lie close to the separation axis seem to be favoured.

To inspect which paths dominate the superposition, consider Figures 5.6 - 5.9, which depict the paths with largest and smallest weights for $r = 3$ and $r = 10$. For $r = 3$, the straight Wilson line has the largest contribution to the overall superposition as its weight, which has an absolute value

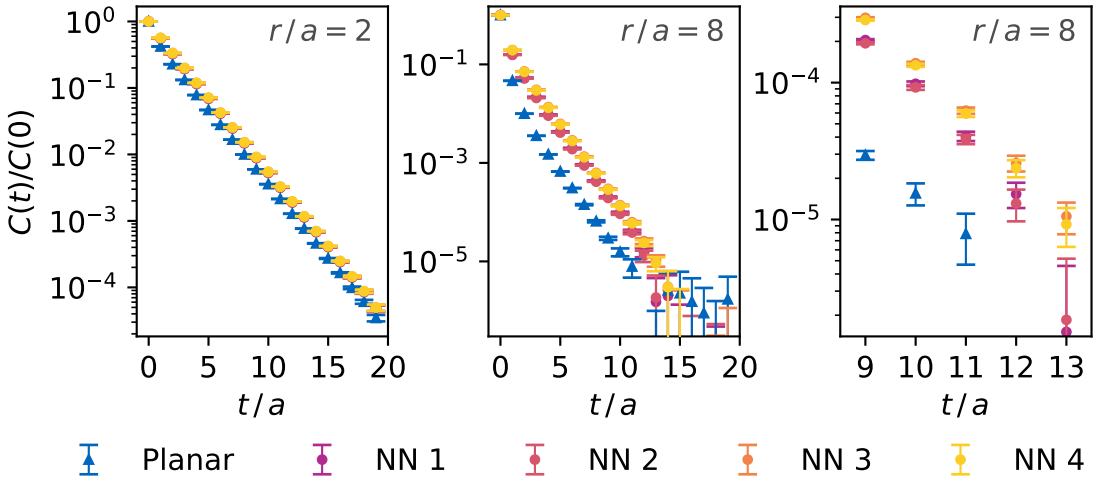


Figure 5.1: Normalised correlators from planar Wilson loops (blue data points) and from the neural network models (purple to yellow data points) for spatial separations $r = 2$ and $r = 8$. The data points of the different neural network models are only partially visible because they overlap.

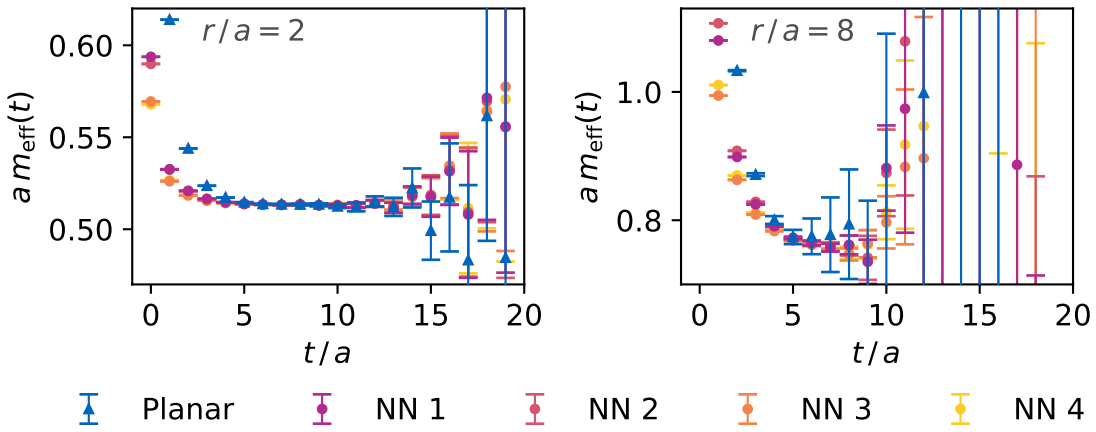


Figure 5.2: Effective masses from planar Wilson loops (blue data points) and from the neural network models (purple to yellow data points) for spatial separations $r = 2$ and $r = 8$.

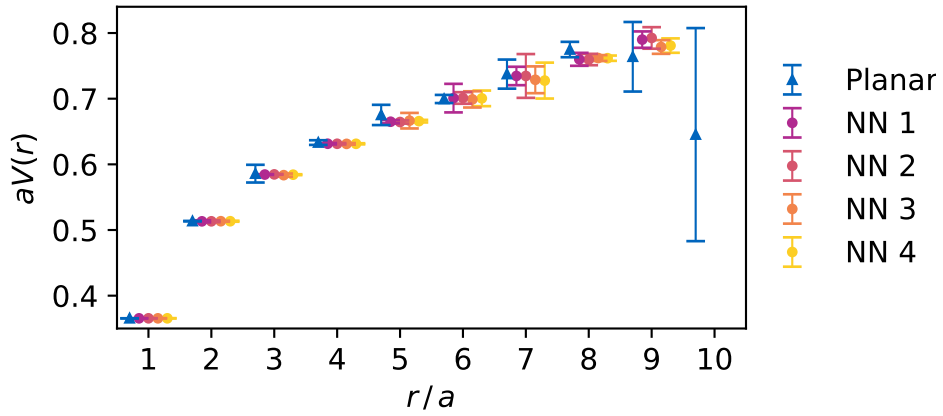


Figure 5.3: The static potential computed from planar Wilson loops and from neural network parametrised Wilson loops. The data points are horizontally offset for better readability.

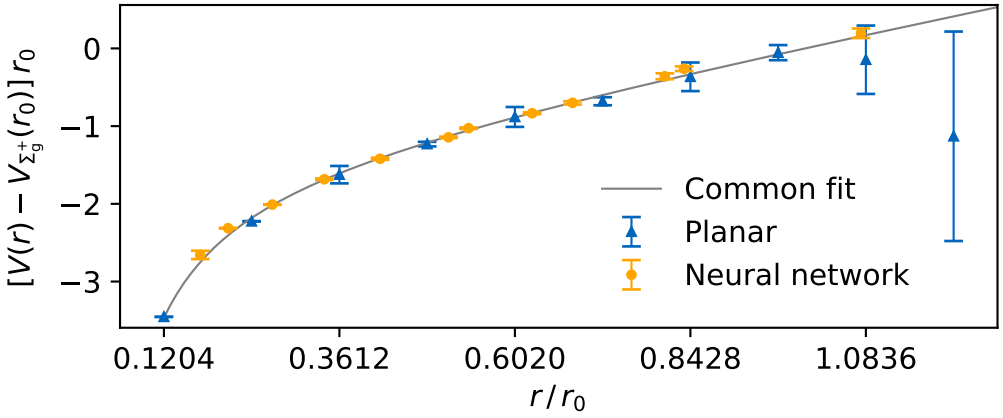


Figure 5.4: The static quark-antiquark potential computed with neural network parametrised Wilson loops for off-axis separations (orange data points) and the static potential obtained from planar Wilson loops for on-axis separations (blue data points). The grey line is the best Cornell potential fit for the joint set of data points.

of 7.657(1) [1], is significantly larger than the second largest weight, which has an absolute value of 4.769(1). The next-largest weights correspond to paths with one plaquette insertion each, where the plaquette lies either in the xz-plane or the yz-plane. Contrary to this, the least contributing paths extend further from the separation axis and have a large total path length. For $r = 10$, the situation is similar, except that here, the straight Wilson line does not give the leading contribution.

In fact, longer paths are systematically suppressed in the superposition, as can be seen in the distribution of the absolute values of the weights, which are shown in Figures 5.10 and 5.11 for $r = 3$ and $r = 10$ respectively. In both cases, there is a large number of paths that do not significantly contribute to the superposition (Note the logarithmic axes). These paths tend to have long path lengths, indicating that they were produced by the convolutional layers. Nevertheless, as is evident from the effective masses, adding a second convolutional layer increased the ground state overlap. We conclude that while individual complicated paths do not significantly contribute to the superposition, these paths collectively influence the ground state overlap of the interpolator.

[1] The weight of each path is computed from the corresponding product of neural network parameters. The weights are therefore complex and not normalised. When comparing weights, we always refer to the weights' absolute values.

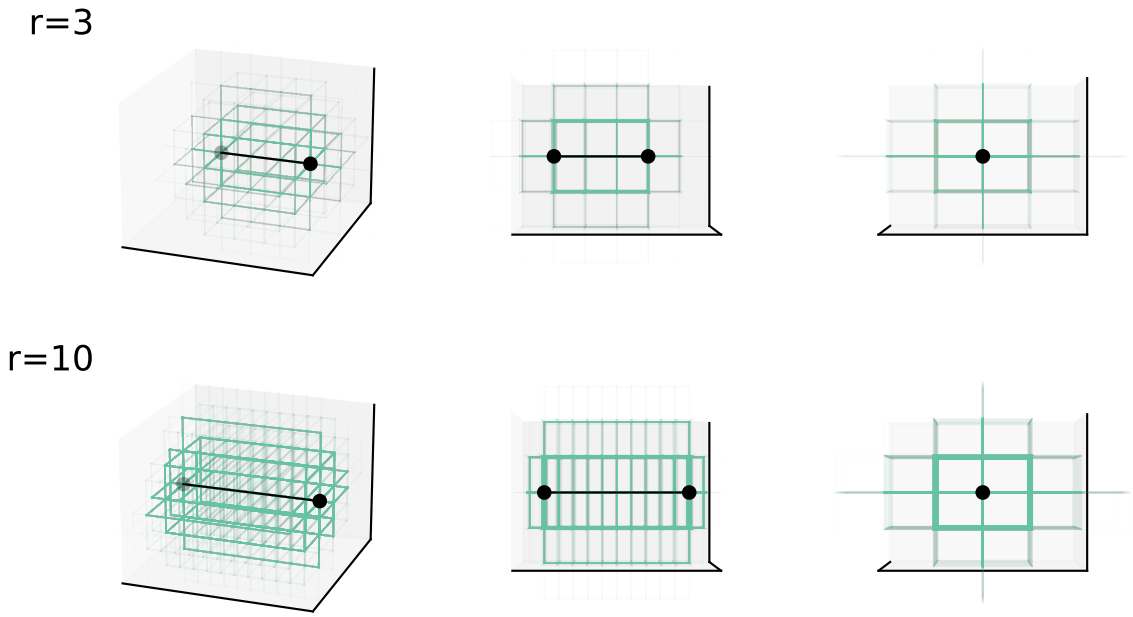


Figure 5.5: Paths that appear in the neural network parametrised superposition for $r = 3$ and $r = 10$ respectively, each shown from three different angles. The opacity of each path is proportional to the absolute value of its associated weight.

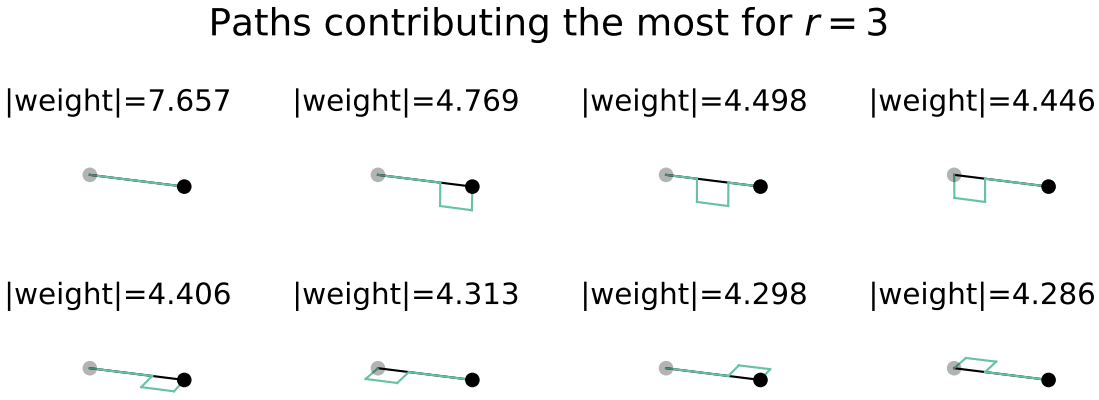


Figure 5.6: Paths with the largest weights for $r = 3$.

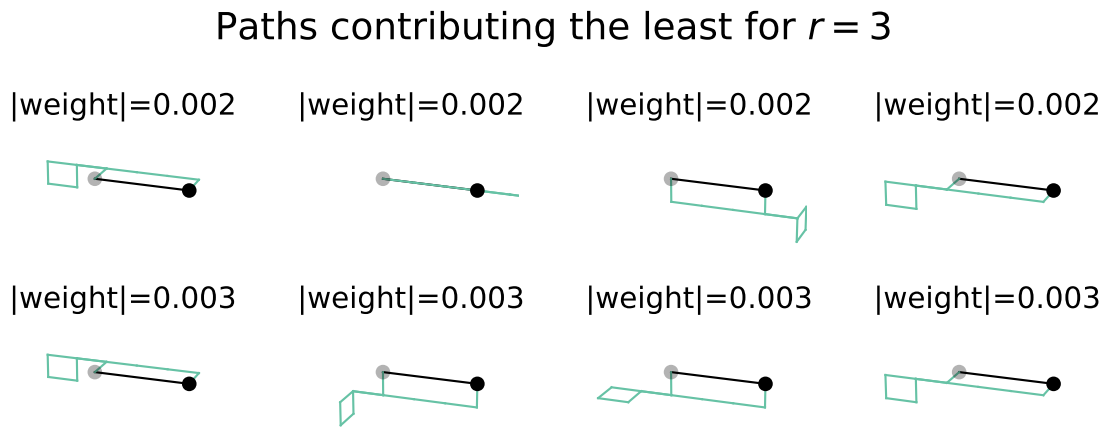


Figure 5.7: Paths with the smallest weights for $r = 3$.

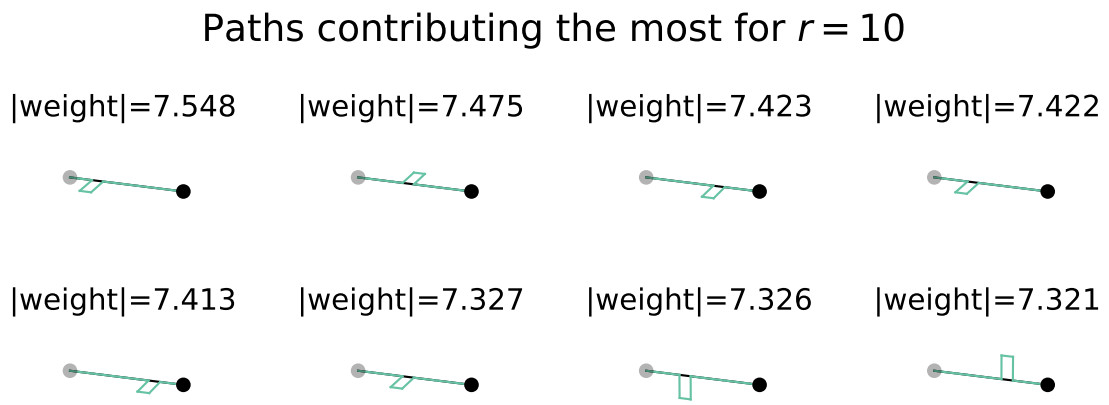


Figure 5.8: Paths with the largest weights for $r = 10$.

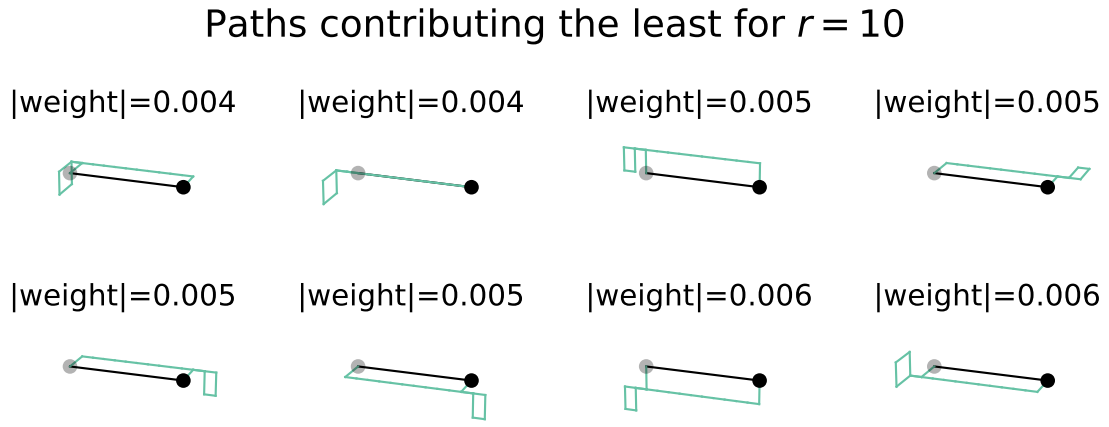


Figure 5.9: Paths with the smallest weights for $r = 10$.

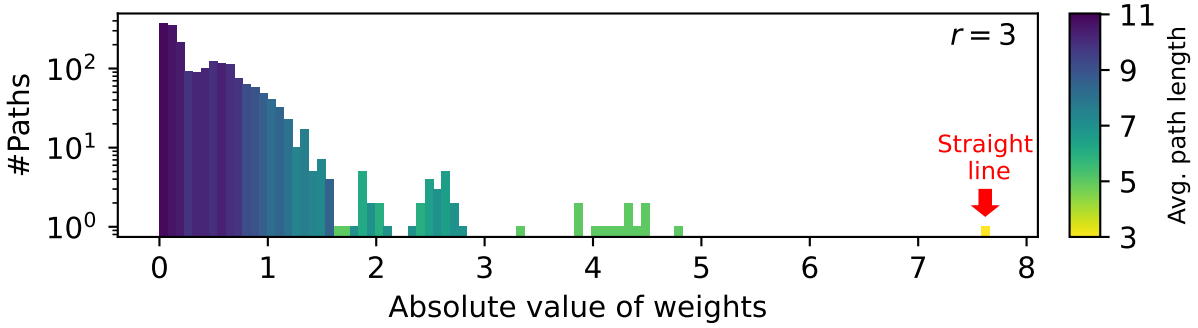


Figure 5.10: Distribution of the absolute values of the weights for $r = 3$, colour-coded by the average length of the paths in each bin.

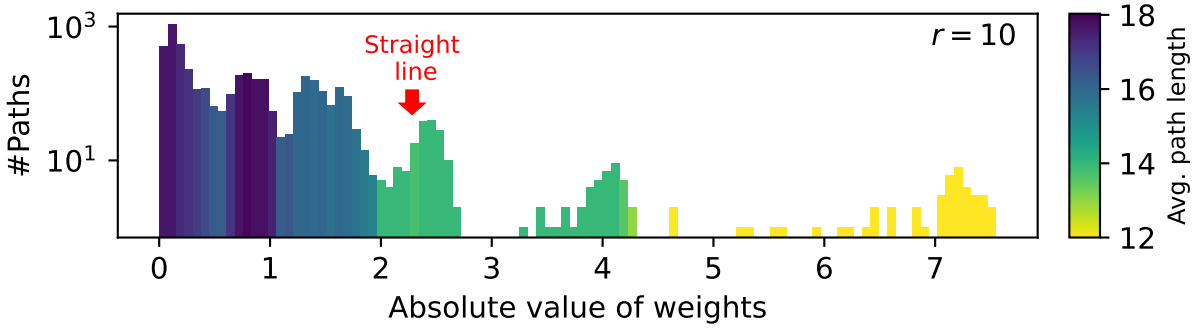


Figure 5.11: Distribution of the absolute values of the weights for $r = 10$, colour-coded by the average length of the paths in each bin.

6 Application to Hybrid States

A modified version of the neural network presented in Chapter 4 may be used to extract the static potentials of quarkonium hybrids. To this end, it must be ensured that the neural network produces a state with the correct Λ_η^ϵ quantum numbers. In this chapter we construct initial states with specified Λ_η^ϵ quantum numbers, we modify our neural network such that it conserves these quantum numbers and we present the static potential we obtain in this way for the Π_u state.

6.1 Basic Concept

Again, we parametrise the Wilson loop by a neural network and write

$$W_{\mathcal{L},r,t} \equiv \mathrm{tr}\left[S(\mathbf{x},\mathbf{y},t)\,T(\mathbf{y},t,t+T)\,S^\dagger(\mathbf{x},\mathbf{y},t+T)\,T^\dagger(\mathbf{x},t,t+T)\right] \tag{6.1}$$

where the temporal transporters remained unmodified as they represent the heavy quark and antiquark. The information on the quantum numbers of the state is contained in the spatial segments S and $S^\dagger$. The neural network attempts to minimise the same loss function as before, under the constraint that the neural network's output has the correct Λ_η^ϵ quantum numbers.

6.2 Notation

We denote the position of the quark by $\mathbf{x}$ and the one of the antiquark by $\mathbf{y}$. In the following, we will identify the operator S with its corresponding path on the lattice, and denote it by the directions of its individual link variables:

$$(x_1, x_2, ..., x_n) \;\equiv\; [x_1, x_2, ..., x_n] \;\equiv\; [x_1, (x_2, ..., x_n)] \;\equiv\; S \tag{6.2}$$

As an example, the path shown in Figure 6.1 is denoted by $(x, y, x, -y, -y, x)$.

If S is a superposition of two paths, we write it as a sum of its constituent paths, e.g. as

$$(x_1, x_2, x_3, x_4) + (x_1, x_5, x_6, x_7) \;=\; [x_1, (x_2, x_3, x_4) + (x_5, x_6, x_7)] \tag{6.3}$$

We demand the path's endpoints to be the position of the quark and antiquark respectively, i.e. we require

$$\sum_i \mathrm{sgn}(x_i)\,\hat{e}_{x_i} \;=\; \mathbf{y} - \mathbf{x}. \tag{6.4}$$

For better readability, we choose the coordinate system such that the quark is at position $\mathbf{x} = (0, 0, +r/2)$ and the antiquark at position $\mathbf{y} = (0, 0, -r/2)$. In our implementation, however, we consider also other configurations.

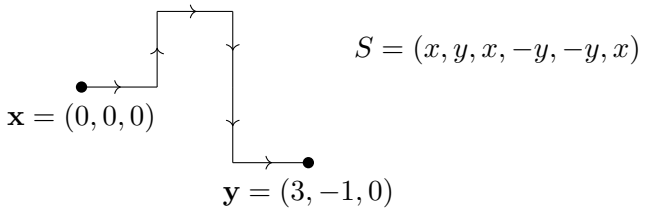


Figure 6.1: Example for how we denote paths from a fixed position $\mathbf{x}$ to a fixed position $\mathbf{y}$.

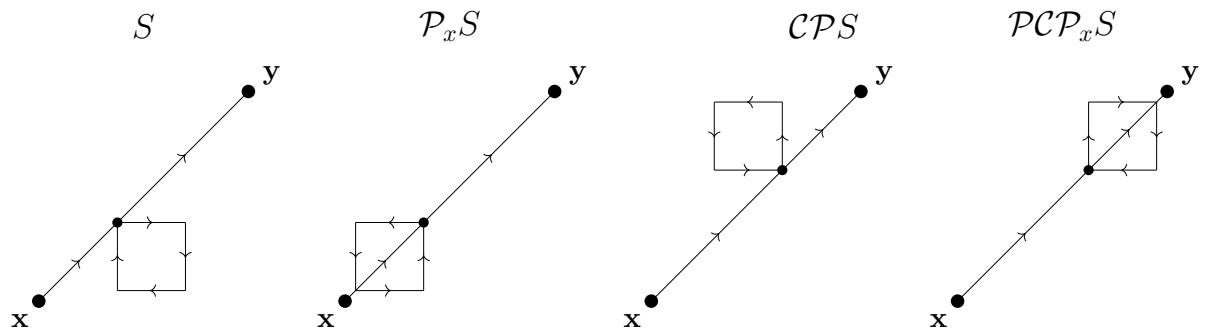


Figure 6.2: The operators $\mathcal{P}_x$, $\mathcal{PC}$, $\mathcal{PCP}_x$ map a path $S = \phi_i^{(0)}$ to another path $\phi_j^{(0)}$.

6.2.1 Action of $\mathcal{C}$, $\mathcal{P}$, $\mathcal{P}_x$ and R on a Path

The linear operators that are needed to characterise quarkonium hybrids are charge conjugation $\mathcal{C}$, parity about the midpoint between quark and antiquark $\mathcal{P}$, reflection over the yz-plane $\mathcal{P}_x$, and rotation by an angle $\frac{k\pi}{2}$ ($k \in \mathbb{N}_0$) around the separation axis $R(\frac{k\pi}{2})$. On a path $(x_1, x_2, ..., x_{n-1}, x_n)$ they act as follows:

- $\mathcal{C}$ reverses the order of the elements of the path as well as the sign of each element: $\mathcal{C}(x_1, x_2, ..., x_{n-1}, x_n) \;=\; (-x_n, -x_{n-1}, ..., -x_2, -x_1)$
- $\mathcal{P}$ reverses the sign of each element: $\mathcal{P}(x_1, x_2, ..., x_{n-1}, x_n) \;=\; (-x_1, -x_2, ..., -x_{n-1}, -x_n)$
- $\mathcal{P}_x$ reverses the sign of all elements that are equal to x or $-x$.
- $R(\frac{n\pi}{2})$ rotates all elements that are not along the separation axis by an angle $\frac{n\pi}{2}$ around the latter and leaves the other elements as well as the order of the elements unchanged.

6.3 Construction of Initial States with Specified Quantum Numbers

We now construct initial states $\tilde{\phi}_i^{(0)}$ that have definite quantum numbers Λ_η^ϵ, where Λ is the angular momentum along the z-axis, ϵ is the respective eigenvalue of $\mathcal{P}_x$, and η is the respective eigenvalue of $\mathcal{PC}$. Inspired by Capitani et al. [4], we construct the $\tilde{\phi}_i^{(0)}$ from states $\phi_i^{(0)}$ that are by themselves not eigenstates of the relevant operators. In our case, the $\phi_i^{(0)}$ are straight spatial lines with one plaquette insertion each.

To ensure that the constructed states have definite angular momentum Λ along the z-axis, consider the continuum state

$$|\psi\rangle \;=\; \int_0^{2\pi} \mathrm{d}\phi \; \exp(i\Lambda\phi)\, R(\phi)\, O_S\, |\Omega\rangle \tag{6.5}$$

where the operator $O_S = \bar{Q}(\mathbf{y}) S(\mathbf{y}, \mathbf{x}) Q(\mathbf{x})$ creates the quark-antiquark pair. $|\psi\rangle$ is an eigenstate of $R(\alpha)$ that corresponds to eigenvalue $e^{-i\Lambda\alpha}$:

$$R(\alpha)\,|\psi\rangle \;=\; \int_0^{2\pi} \mathrm{d}\phi \; \exp(i\Lambda(\phi-\alpha)) R(\phi)\, O_\mathrm{S}\, |\Omega\rangle \;=\; e^{-i\Lambda\alpha}\,|\psi\rangle \tag{6.6}$$

Since the angular momentum operator J_z is the generator of rotations around the z-axis, it holds that $R(\alpha)\,|\psi\rangle = e^{-iJ_z\alpha}\,|\psi\rangle$. Hence, $e^{-iJ_z\alpha}\,|\psi\rangle = e^{-i\Lambda\alpha}\,|\psi\rangle$, which means that $|\psi\rangle$ is a state of definite z-component of angular momentum Λ.

The corresponding lattice expression is

$$|\psi\rangle \;=\; \sum_{k=0}^{3} \exp\!\left(\frac{i\pi\Lambda k}{2}\right) R\left(\frac{\pi k}{2}\right) O_{\mathrm{S}}\,|\Omega\rangle \;\equiv\; A O_{\mathrm{S}}\,|\Omega\rangle \tag{6.7}$$

where O_S is now the operator corresponding to path S from the quark's position to the antiquark's position. The operators Q and $\bar{Q}$ on the lattice correspond to the temporal transporters, which are not affected by the action of $\mathcal{P}_x, \mathcal{PC}, \mathcal{PCP}_x$ and R, and therefore not considered in the following.

In order to fix also the eigenvalues ϵ and η, we apply the projectors $\mathbb{P}_{\mathcal{PC}} = \frac{1}{2}(1 + \eta\,\mathcal{PC})$ and $\mathbb{P}_{\mathcal{P}_x} = \frac{1}{2}(1 + \epsilon\,\mathcal{P}_x)$ to $A\,\phi_i^{(0)}$, and use the result as input to our neural network:

$$\tilde{\phi}_i^{(0)} \;=\; \frac{1}{4}\left(1 + \epsilon\,\mathcal{P}_x + \eta\,\mathcal{PC} + \eta\epsilon\,\mathcal{PCP}_x\right) A\,\phi_i^{(0)} \tag{6.8}$$

In this way we construct states with the correct quantum numbers from the $\phi_i^{(0)}$, which are relatively simple objects.

6.4 Propagation of Quantum Numbers through the Neural Network

Now that we have constructed initial states with the correct quantum numbers, we need to ensure that the neural network preserves these quantum numbers, i.e. we want to achieve that

$$O\,S(\tilde{\phi}_i^{(0)}) \;=\; S(O\,\tilde{\phi}_i^{(0)}) \;=\; S(o\,\tilde{\phi}_i^{(0)}) \;=\; o\,S(\tilde{\phi}_i^{(0)})\;\forall\, O \in \{\,\mathcal{P}_x, \mathcal{PC}, \mathcal{PCP}_x, R\,\} \tag{6.9}$$

where S is the neural network parametrised spatial line and $o \in \mathbb{C}$ an eigenvalue of O.

As illustrated in Figure 6.2, each of the $\phi_i^{(0)}$ is mapped to another element of $\{\,\phi_j^{(0)}\,\}$ by the operators $\mathcal{P}_x$, $\mathcal{PC}$, and $\mathcal{PCP}_x$. The same is true for the rotation operator R. We write

$$O\,\phi_i^{(0)} \;=\; \phi_{O(i)}^{(0)} \quad \text{for} \;\; O \in \{\,\mathcal{P}_x, \mathcal{PC}, \mathcal{PCP}_x, R\,\}\,. \tag{6.10}$$

6.4.1 Action of $\mathcal{C}$, $\mathcal{P}$, $\mathcal{P}_x$ and R on the Output of a Neural Network Layer

In order to construct a neural network that satisfies (6.9), we need to understand how the operators $\mathcal{P}_x$, $\mathcal{PC}$, $\mathcal{PCP}_x$, and R act on the output of each neural network layer. As we are not interested in finding the quarkonium ground state, we will not include bias terms in the layers.

Linear Layer

For any linear operator O it holds that

$$O\,\phi_i^{n+1} \;=\; O\,\omega_{ij}\phi_j^n \;=\; \omega_{ij}\,O\,\phi_j^n \tag{6.11}$$

i.e. the operator O simply acts on the input of the layer.

Convolutional Layer

The convolutional layer without bias term is given by

$$\begin{aligned} \phi_i^{(n+1)} &= \omega_{ij\mu}^{(n)}\, U_\mu(x)\, \phi_j^{(n)}(x+\hat{\mu}, y+\hat{\mu})\, U_\mu^\dagger(y) \\ &= \omega_{ij\mu}^{(n)}\, [\,\mu, \phi_j^{(n)}, -\mu\,] \;\equiv\; \phi_i^{(n+1)}(\phi^{(n)}, \omega) \end{aligned} \tag{6.12}$$

where we suppressed the arguments of $\phi_i^{(n+1)}$ in our path notation, as the path is specified solely by the directions of its individual links.

The relevant operators act on the $\phi_i^{(n+1)}$ as follows:

$$\begin{aligned} \mathcal{P}_x\, \phi_i^{(n+1)} &= \omega_{ij\mu}^{(n)} \left(\delta_{x|\mu|} [\,-\mu, \mathcal{P}_x\, \phi_j^{(n)}, \mu\,] + (1-\delta_{x|\mu|}) [\,\mu, \mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \right) \\ &= \omega_{ij-\mu}^{(n)}\, \delta_{x|\mu|} [\,\mu, \mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] + \omega_{ij\mu}^{(n)} (1-\delta_{x|\mu|}) [\,\mu, \mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \\ &= \phi_i^{(n+1)}(\mathcal{P}_x\, \phi^{(n)}, \omega^{\mathcal{P}_k}) \\ &\text{where} \quad \omega_{ij\mu}^{\mathcal{P}_x} = \omega_{ij-\mu}^{(n)}\, \delta_{x|\mu|} + \omega_{ij\mu}^{(n)} (1-\delta_{x|\mu|}); \end{aligned}$$

$$\begin{aligned} \mathcal{P}\mathcal{C}\, \phi_i^{(n+1)} &= \omega_{ij\mu}^{(n)}\, [\,-\mu, \mathcal{P}\mathcal{C}\, \phi_j^{(n)}, \mu\,] \\ &= \omega_{ij-\mu}^{(n)}\, [\,\mu, \mathcal{P}\mathcal{C}\, \phi_j^{(n)}, -\mu\,] \\ &= \phi_i^{(n+1)}(\mathcal{P}\mathcal{C}\, \phi^{(n)}, \omega_{ij\mu}^{\mathcal{P}\mathcal{C}}) \\ &\text{where} \quad \omega_{ij\mu}^{\mathcal{P}\mathcal{C}} = \omega_{ij-\mu}^{(n)}; \end{aligned}$$

$$\begin{aligned} \mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi_i^{(n+1)} &= \mathcal{P}\mathcal{C} \left(\omega_{ij-\mu}^{(n)}\, \delta_{x|\mu|} [\,\mu, \mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \;+\; \omega_{ij\mu}^{(n)} (1-\delta_{x|\mu|}) [\,\mu, \mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \right) \\ &= \omega_{ij-\mu}^{(n)}\, \delta_{x|\mu|} [\,-\mu, \mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi_j^{(n)}, \mu\,] \;+\; \omega_{ij\mu}^{(n)} (1-\delta_{x|\mu|}) [\,-\mu, \mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi_j^{(n)}, \mu\,] \\ &= \omega_{ij\mu}^{(n)}\, \delta_{x|\mu|} [\,\mu, \mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \;+\; \omega_{ij-\mu}^{(n)} (1-\delta_{x|\mu|}) [\,\mu, \mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi_j^{(n)}, -\mu\,] \\ &= \phi_i^{(n+1)}(\mathcal{P}\mathcal{C}\mathcal{P}_x\, \phi^{(n)}, \omega^{\mathcal{P}\mathcal{C}\mathcal{P}_x}) \\ &\text{where} \quad \omega^{\mathcal{P}\mathcal{C}\,\mathcal{P}_k} = \omega_{ij\mu}^{(n)}\, \delta_{k|\mu|} + \omega_{ij-\mu}^{(n)} (1-\delta_{k|\mu|}); \end{aligned}$$

$$\begin{aligned} R\, \phi_i^{(n+1)} &= \omega_{ij\mu}^{(n)}\, [\, R\mu, R\phi_j^{(n)}, -R\mu\,] \\ &= \omega_{ijR\mu}^{(n)}\, [\,\mu, R\phi_j^{(n)}, -\mu\,] \\ &= \phi_i^{(n+1)}(R\phi^{(n)}, \omega^R) \\ &\text{where} \quad \omega_{ij\mu}^{R} = \omega_{ijR(\mu)} \end{aligned}$$

Constraints for the Neural Network's Parameters

In order for the Λ_η^ϵ quantum numbers to be preserved by the neural network, we must have $\omega_{ij\mu}^O = \omega_{ij\mu}^{(n)}$ at each layer and for each of the operators $O \in \{\, \mathcal{P}_x, \mathcal{P}\mathcal{C}, \mathcal{P}\mathcal{C}\mathcal{P}_x, R\,\}$. According to

(6.11), this is automatically satisfied for the linear layer. Contrary to this, the parameters of the convolutional layer are constrained by $\omega_{ij\mu} = \omega_{ij-\mu}$ due to the above considerations for the action of $\mathcal{P}_x$, $\mathcal{PC}$, and $\mathcal{PCP}_x$ on $\phi_i^{(n+1)}$. In addition, because of the action of $R(\pi/2)$ on $\phi_i^{(n+1)}$, we require $\omega_{ijx} = \omega_{ijy}$. These constraints ensure that, if the $\phi_i^{(n)}$ are eigenstates of O corresponding to eigenvalue o, than so are the $\phi_i^{(n+1)}$:

$$O\,\phi_i^{(n+1)}(\phi^{(n)}) \;=\; \phi_i^{(n+1)}(O\,\phi^{(n)}) \;=\; \phi_i^{(n+1)}(o\,\phi^{(n)}) \;=\; o\,\phi_i^{(n+1)}(\phi^{(n)}) \tag{6.13}$$

This holds for linear and convolutional layers, but not for bilinear layers. For the latter one instead arrives at $O\,\phi_i^{(n+1)}(\phi^{(n)}) = o^2\,\phi_i^{(n+1)}(\phi^{(n)})$, which means that the bilinear layer maps states of eigenvalue $o = -1$ to the eigenspace corresponding to eigenvalue $+1$, and states of eigenvalues $o = \pm i$ to the eigenspace corresponding to eigenvalue -1. This is undersirable, as the $\tilde{\phi}_i^{(0)}$ are eigenstates of the rotation operator $R(\frac{n\pi}{2})$ corresponding to eigenvalues $e^{-\frac{in\pi}{2}} = \{\pm 1, \pm i\}$. Therefore, bilinear layers are not suitable for the treatment of hybrid states. Instead, we define a trilinear layer.

Trilinear Layer

The defining equation for the trilinear layer reads

$$\phi_i^{(n+1)} \;= \sum_{j,k,l} \omega_{ijkl}^{(n)}\,\phi_j^{(n)}\phi_k^{(n)\dagger}\phi_l^{(n)} \tag{6.14}$$

$$= \omega_{ijkl}^{(n)}\,[\phi_j^{(n)}, \phi_k^{(n)\dagger}, \phi_l^{(n)}] \;\equiv\; \phi_i^{(n+1)}(\phi^{(n)}, \omega) \tag{6.15}$$

where $\omega_{ijkl}^{(n)} \in \mathbb{C}$. $\mathcal{P}_x$, $\mathcal{PC}$, $\mathcal{PCP}_x$, and R act on the output of the trilinear layer as follows:

$$\mathcal{P}_x\,\phi_i^{(n+1)} \;=\; \phi_i^{(n+1)}(\mathcal{P}_x\,\phi^{(n)}, \omega)$$

$$\begin{aligned}
\mathcal{PC}\,\phi_i^{(n+1)} \;&=\; \omega_{ijkl}^{(n)}\,[\mathcal{PC}\phi_l^{(n)}, \mathcal{PC}\phi_k^{(n)\dagger}, \mathcal{PC}\phi_j^{(n)}] \\
&=\; \omega_{ilkj}^{(n)}\,[\mathcal{PC}\phi_l^{(n)}, \mathcal{PC}\phi_k^{(n)\dagger}, \mathcal{PC}\phi_j^{(n)}] \\
&=\; \phi_i^{(n+1)}(\mathcal{PC}\,\phi^{(n)}, \omega^{\mathcal{PC}}) \\
&\text{where} \quad \omega_{ijkl}^{\mathcal{PC}} \;=\; \omega_{ilkj}^{(n)};
\end{aligned}$$

$$\begin{aligned}
\mathcal{PCP}_x\,\phi_i^{(n+1)} \;&=\; \mathcal{PC}\left(\omega_{ijkl}^{(n)}\,[\mathcal{P}_x\phi_j^{(n)}, \mathcal{P}_x\phi_k^{(n)\dagger}, \mathcal{P}_x\phi_l^{(n)}]\right) \\
&=\; \omega_{ijkl}^{(n)}\,[\mathcal{PCP}_x\,\phi_l^{(n)}, \mathcal{PCP}_x\,\phi_k^{(n)\dagger}, \mathcal{PCP}_x\,\phi_j^{(n)}] \\
&=\; \phi_i^{(n+1)}(\,\mathcal{PCP}_x\,\phi^{(n)}, \omega^{\mathcal{PCP}_x}) \\
&\text{where} \quad \omega_{ijkl}^{\mathcal{PCP}_k} \;=\; \omega_{ilkj}^{(n)};
\end{aligned}$$

$$\begin{aligned}
R\,\phi_i^{(n+1)} \;&=\; \omega_{jklmp}^{(n)}\,[R\phi_j^{(n)}, R\phi_k^{(n)\dagger}, R\phi_l^{(n)}] \\
&=\; \phi_i^{(n+1)}(R\,\phi^{(n)}, \omega)
\end{aligned}$$

Model name	Terms included in loss function	Layer setup	# Channels
NN 5	"tower of exponentials"	1× Linear, 1× Trilinear 2× Convolutional	$x \to 5$ $5 \to 5$ $5 \to 5 \to 1$

Table 6.1: Model used for the computation of the Π_u potential.

As we require $\omega^O_{ijkl} = \omega^{(n)}_{ijkl}$, the parameters of the trilinear layer are constrained by $\omega_{ijkl} = \omega_{ilkj}$. These constraints guarantee that the quantum numbers correctly propagate through the layer:

$$
\begin{aligned}
O\,\phi_i^{(n+1)}(\phi^{(n)}) &= \phi_i^{(n+1)}(O\,\phi^{(n)}) && (6.16)\\
&= \phi_i^{(n+1)}(o\,\phi^{(n)}) && (6.17)\\
&= \underbrace{o^\dagger o}_{1}\, o\,\phi_i^{(n+1)}(\phi^{(n)}) && (6.18)\\
&= o\,\phi_i^{(n+1)}(\phi^{(n)}) \quad \forall\, O \in \{\,\mathcal{P}_x, \mathcal{PC}, \mathcal{PCP}_x, R\,\} && (6.19)
\end{aligned}
$$

6.5 Layer Setup

As the results for the ground state were notably improved by a second convolutional layer, we use two convolutional layers also in the case of the hybrid potentials. The layer setup we applied to find the Π_u potential is described in Table 6.1.

6.6 Results

The Π_u potential computed with model NN 5 and the Σ_g^+ potential computed with model NN 4 are plotted in Figure 6.3, with fits (2.32) and (2.33) respectively. While the uncertainties in the Π_u potential computed with model NN 5 are very large, comparison to Figure 2.3 shows that the data points lie in the correct interval for the Π_u potential.

The main source of the uncertainties lies in the noise in the effective masses. The latter are plotted in Figure 6.4 for the separations $r = 2$ and $r = 8$. Clearly, the uncertainties in the effective mass become large very quickly, rendering the determination of the static potential very difficult. A more involved layer structure or smearing might help increase the signal-to-noise ratio, thereby reducing the uncertainties in the static potential.

We conclude that neural network parametrised Wilson loops are, in principle, suitable for the determination of hybrid potentials, but that the method has to be refined further in order to yield meaningful results.

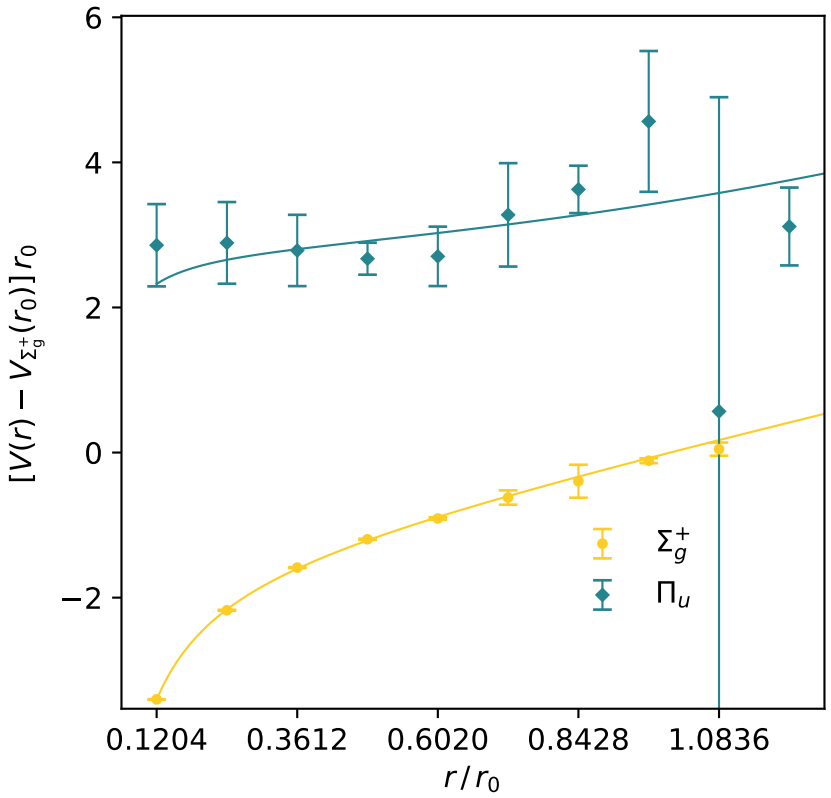


Figure 6.3: Static potentials computed from neural network parametrised Wilson loops for the Σ_g^+ and Π_u states. The solid lines are fits corresponding to the functions (2.32) and (2.33) respectively.

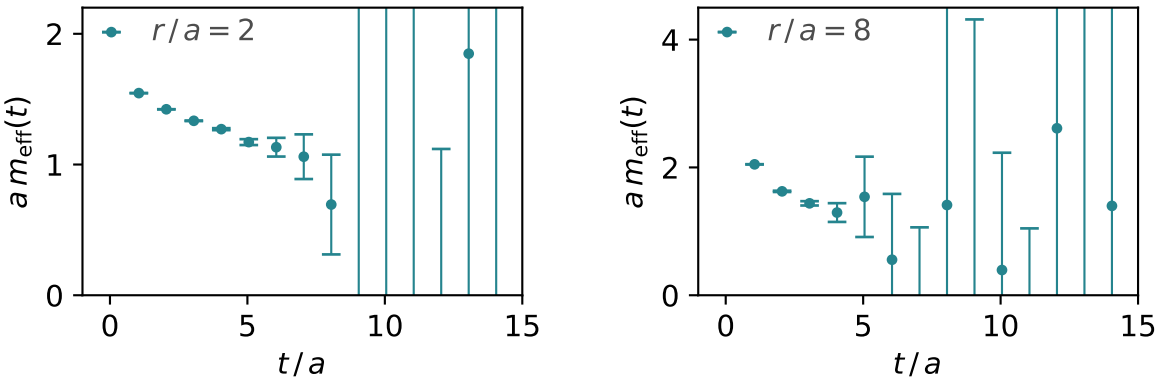


Figure 6.4: Effective mass for the Π_u state as computed with neural network model NN 4.

7 Outlook and Conclusion

In this work we parametrised the Wilson using a neural network and demonstrated that this technique can be used to extract the static potentials of the quarkonium ground state and of quarkonium hybrid states. Compared to planar Wilson loops, the neural network parametrised Wilson loops lead to earlier effective mass plateaus, corresponding to a larger overlap with the state under consideration. This allows for a cleaner determination of the height of the effective mass plateau, reflecting itself in smaller uncertainties in the extracted static potential at large quark-antiquark separations.

In the case of the hybrid state Π_u, we could compute the static potential using neural network parametrised Wilson loops, but only with large uncertainties. It remains to be investigated whether a more involved neural network structure would lead smaller uncertainties.

Altogether, our method is a promising alternative for the computation of the static potentials of quarkonium and quarkonium hybrids and a proof of concept for the optimisation of lattice interpolators through a neural network. This opens up several questions that might be addressed in future research:

- **More hybrid states**: We laid out a procedure for the computation of the static potential of a general Λ_η^ϵ quarkonium state, and successfully applied it to the Σ_g^+ and Π_u states. This procedure remains to be applied to further hybrid states.

- **More complex layer setups**: From our results for the non-exotic Σ_g^+ state, we see that the complexity of the neural network is decisive for the achievement of early effective mass plateaus. It is, therefore, desirable to test neural networks with large numbers of convolutional layers and/or multilinear layers. Additionally, one could increase the variety of paths considered by the neural network by increasing the number of output channels of any given layer.

- **Off-axis separations for hybrid states**: We performed the computation for hybrid states only for on-axis separations. In order to treat off-axis separations, one would have to construct corresponding input paths that satisfy the symmetry requirements of the respective quantum numbers. This would increase the spatial resolution of the static hybrid potentials.

- **(Exponential) activation function**: In machine learning, it is common to apply activation functions such as ReLu or a sigmoid function in between the layers. The effect of such activation functions could be investigated. In particular, it might be interesting to consider the effect of an exponential activations function that projects the superposition of different paths onto SU(3). To this end, one could define locally transforming objects

$$\tilde{\phi}_i^{(n)} \;\equiv\; \phi_i^{(n)}\phi^{(0)\dagger} \;\to\; \big(\Omega(x)\,\phi_i^{(n)}\,\Omega^\dagger(y)\big)\big(\Omega(x)\,\phi^{(0)}(x,y)\,\Omega^\dagger(y)\big)^\dagger \;=\; \Omega(x)\,\tilde{\phi}_i^{(n)}\,\Omega^\dagger(x) \tag{7.1}$$

  and pass them through an exponential activation function:

$$\phi_i^{(n+1)} = \exp\left[\tilde{\phi}_i^{(n)} - \frac{1}{3}\,\mathrm{tr}\Big(\tilde{\phi}_i^{(n)}\Big)\mathbb{1} - \tilde{\phi}_i^{(n)\dagger} + \frac{1}{3}\,\mathrm{tr}\Big(\tilde{\phi}_i^{(n)\dagger}\Big)\mathbb{1}\right]\phi^{(0)} \tag{7.2}$$

  By construction, the exponent is anti-Hermitian and traceless, i. e. it is of the form iX where $X \in \mathfrak{su}(3)$. Therefore, given that $\phi^{(0)} \in \mathrm{SU}(3)$, the output of the exponential layer is also an element of SU(3).

- **Plaquettes in temporal direction**: The input to our neural network consists of straight Wilson lines with one plaquette insertion each, where the plaquette contains only spatial links.

One could consider the effect of adding plaquette-inserted Wilson loops with plaquettes that contain temporal links.

- **Continuum limit**: Thus far we have worked exclusively with a $20^3 \times 40$ lattice. The computations should be extended to larger lattices, corresponding to smaller lattice spacing a. This is necessary for taking the continuum limit $a \to 0$.

Many of these possible improvements would, however, increase the memory demands of the method, which is currently the limiting factor for our computations.

We conclude that neural networks can be used to optimise the Wilson loop towards a larger overlap with the quarkonium ground state or quarkonium-related hybrid states, allowing for the computation of the corresponding static potentials. This opens up interesting opportunities for future research, particularly concerning the treatment of hybrid states that have not yet been investigated in this thesis.

References

[1] M. Gell-Mann. 'A schematic model of baryons and mesons'. In: *Physics Letters* 8.3 (1964), pp. 214–215. ISSN: 0031-9163. DOI: https://doi.org/10.1016/S0031-9163(64)92001-3.

[2] S.-K. Choi et al. 'Observation of a Narrow Charmoniumlike State in Exclusive $B^{\pm} \to K^{\pm}\pi^{+}\pi^{-}J/\psi$ Decays'. In: *Phys. Rev. Lett.* 91 (26 Dec. 2003), p. 262001. DOI: 10.1103/PhysRevLett.91.262001.

[3] S. Navas et al. 'Review of particle physics'. In: *Phys. Rev. D* 110.3 (2024), p. 030001. DOI: 10.1103/PhysRevD.110.030001.

[4] Stefano Capitani et al. 'Precision computation of hybrid static potentials in SU(3) lattice gauge theory'. In: *Physical Review D* 99.3 (Feb. 2019). ISSN: 2470-0029. DOI: 10.1103/physrevd.99.034502.

[5] Carolin Schlosser and Marc Wagner. 'Hybrid static potentials in SU(3) lattice gauge theory at small quark-antiquark separations'. In: *Physical Review D* 105.5 (Mar. 2022). ISSN: 2470-0029. DOI: 10.1103/physrevd.105.054503.

[6] L. M. Brown. *Feynman's Thesis — A New Approach to Quantum Theory*. World Scientific, 2005. DOI: 10.1142/5852.

[7] J. J. Sakurai and J. Napolitano. *Modern Quantum Mechanics*. 3rd ed. Cambridge University Press, 2020.

[8] G. Zweig. 'An SU(3) model for strong interaction symmetry and its breaking. Version 2'. In: *Developments in the quark theory of hadrons. Vol. 1. 1964 - 1978*. Ed. by D. B. Lichtenberg and Simon Peter Rosen. Feb. 1964, pp. 22–101.

[9] O. W. Greenberg. 'Spin and Unitary-Spin Independence in a Paraquark Model of Baryons and Mesons'. In: *Phys. Rev. Lett.* 13 (20 Nov. 1964), pp. 598–602. DOI: 10.1103/PhysRevLett.13.598.

[10] Y. Nambu. 'A systematics of hadrons in subnuclear physics'. In: *Broken Symmetry*, pp. 192–201. DOI: 10.1142/9789812795823_0018.

[11] M. Y. Han and Y. Nambu. 'Three-Triplet Model with Double SU(3) Symmetry'. In: *Phys. Rev.* 139 (4B Aug. 1965), B1006–B1010. DOI: 10.1103/PhysRev.139.B1006.

[12] G. Eichmann. *QCD lecture notes*. URL: http://cftp.ist.utl.pt/~gernot.eichmann/2014-hadron-physics/ (visited on 04/06/2024).

[13] M.E. Peskin and D. V. Schroeder. *An Introduction to Quantum Field Theory*. Westview Press, 1995. ISBN: 0201503972.

[14] E. Eichten et al. 'Charmonium: The model'. In: *Phys. Rev. D* 17 (11 June 1978), pp. 3090–3117. DOI: 10.1103/PhysRevD.17.3090.

[15] Nora Brambilla. 'pNRQCD: Concepts and applications'. In: *4th Workshop on Continuous Advances in QCD*. May 2000, pp. 314–325. arXiv: hep-ph/0008279.

[16] Gunnar S. Bali and Antonio Pineda. 'QCD phenomenology of static sources and gluonic excitations at short distances'. In: *Phys. Rev. D* 69 (2004), p. 094001. DOI: 10.1103/PhysRevD.69.094001.

[17] Nora Brambilla et al. 'The XYZ states: Experimental and theoretical status and perspectives'. In: *Physics Reports* 873 (2020). The XYZ states: experimental and theoretical status and perspectives, pp. 1–154. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2020.05.001.

[18] C. Gattringer and C. B. Lang. *Quantum Chromodynamics on the Lattice. An Introductory Presentation*. Springer, 2010. ISBN: 978-3-642-01849-7. DOI: https://doi.org/10.1007/978-3-642-01850-3.

[19] Dirk Schlingemann. 'From Euclidean field theory to quantum field theory'. In: *Reviews in Mathematical Physics* 11.09 (1999), pp. 1151–1178. DOI: 10.1142/S0129055X99000362.

[20] K. G. Wilson. 'Confinement of quarks'. In: *Physical Review D* (1974). DOI: https://doi.org/10.1103/PhysRevD.10.2445.

[21] K. Symanzik. 'Continuum limit and improved action in lattice theories: (II). O(N) non-linear sigma model in perturbation theory'. In: *Nuclear Physics B* 226.1 (1983), pp. 205–227. ISSN: 0550-3213. DOI: https://doi.org/10.1016/0550-3213(83)90469-8.

[22] Roman Höllwieser et al. 'Constructing static quark-antiquark creation operators from Laplacian eigenmodes'. In: *Phys. Rev. D* 107 (3 Feb. 2023), p. 034511. DOI: 10.1103/PhysRevD.107.034511.

[23] R. Sommer. 'A new way to set the energy scale in lattice gauge theories and its application to the static force and σ_S in SU (2) Yang-Mills theory'. In: *Nuclear Physics B* 411.2–3 (Jan. 1994), pp. 839–854. ISSN: 0550-3213. DOI: 10.1016/0550-3213(94)90473-1.

[24] H. Rothe. *Lattice Gauge Theories*. World Scientific Publishing Company, 2012. ISBN: 9789814365871. DOI: 10.1142/8229.

[25] *The MILC code (version: 7.7.11)*. URL: https://web.physics.utah.edu/~detar/milc/milcv7.html (visited on 10/07/2024).

[26] Nora Brambilla et al. 'Static force from generalized Wilson loops on the lattice using the gradient flow'. In: *Phys. Rev. D* 109 (11 June 2024), p. 114517. DOI: 10.1103/PhysRevD.109.114517. URL: https://link.aps.org/doi/10.1103/PhysRevD.109.114517.

[27] Silvia Necco and Rainer Sommer. 'The $N_f = 0$ heavy quark potential from short to intermediate distances'. In: *Nuclear Physics B* 622.1 (2002), pp. 328–346. ISSN: 0550-3213. DOI: https://doi.org/10.1016/S0550-3213(01)00582-X.

[28] A. Francis et al. 'Critical point and scale setting in SU(3) plasma: An update'. In: *Phys. Rev. D* 91 (9 May 2015), p. 096002. DOI: 10.1103/PhysRevD.91.096002. URL: https://link.aps.org/doi/10.1103/PhysRevD.91.096002.

[29] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[30] *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*. ACM, Apr. 2024. DOI: 10.1145/3620665.3640366.

[31] Matteo Favoni et al. 'Lattice Gauge Equivariant Convolutional Neural Networks'. In: *Physical Review Letters* 128.3 (Jan. 2022). ISSN: 1079-7114. DOI: 10.1103/physrevlett.128.032003.

[32] B. Efron and C. Stein. 'The Jackknife Estimate of Variance'. In: *The Annals of Statistics* 9.3 (1981), pp. 586–596. DOI: 10.1214/aos/1176345462.

[33] John D. Hunter. 'Matplotlib: A 2D Graphics Environment'. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[34] Stéfan van der Walt, S Chris Colbert and Gaël Varoquaux. 'The NumPy Array: A Structure for Efficient Numerical Computation'. In: *Computing in Science &; Engineering* 13.2 (Mar. 2011), pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/mcse.2011.37.

A Code for the Neural Network

To make the description of our neural network in Section 4.4 clearer, we provide example code for characteristic parts of the neural network. Note that this does not cover the complete code used in this thesis. For example, for each layer, there are several different implementations corresponding, to on-axis vs. off-axis separations or ground state vs. hybrid state computations. The code provided below should, however, clarify the general structure of these implementations.

The code we provide for the layers corresponds to an implementation for hybrid states as discussed in Chapter 6.

We make extensive use of the PyTorch machine learning framework [30], as well as the software packages matplotlib [33] and numpy [34].

A.1 Neural Network

```
import torch

class NN_example(torch.nn.Module):

    def __init__(self, r, n_deep_nodes, Lambda, epsilon, eta):
        super().__init__()
        self.r = r
        self.n_deep_nodes = n_deep_nodes
        self.Lambda = Lambda
        self.epsilon = epsilon
        self.eta = eta
        self.channels_in = 24 + 16 * r
        self.linear_1 = linear(self.channels_in,n_deep_nodes)
        self.trilinear_1 = trilinear(n_deep_nodes, n_deep_nodes)
        self.conv_1 = convolutional(n_deep_nodes, n_deep_nodes)
        self.conv_2 = convolutional(n_deep_nodes, 1)

    def forward(self, config, twlines, phi, direc):
        x = self.projection(phi, r=r, main=direc)
        x = self.linear_1(x)
        x = self.trilinear_1(x)
        x = self.conv_1(config, x, direc, self.r)
        x = self.conv_2(config, x, direc, self.r)
        corr = temporal_correlation.correlations(config, twlines, x[...,0,:,:],
                                                 direc, self.r, 20)
        return corr
```

Linear Layer

```
import torch

class linear(torch.nn.Module):
    def __init__(self, channels_in, channels_out, f_data_type=torch.float64):
        super().__init__()
        self.weights = torch.nn.Parameter(torch.empty((channels_out,channels_in),
                                                      dtype=f_data_type))
        self.weights_i = torch.nn.Parameter(torch.empty((channels_out,channels_in),
                                                        dtype=f_data_type))
        torch.nn.init.normal_(self.weights, std=0.3)
```

```
        torch.nn.init.normal_(self.weights_i, std=0.3)
    def forward(self, phi):
        return torch.einsum("ij,...jab->...iab", self.weights +
                            1j*self.weights_i, phi)
```

Trilinear Layer

```
import torch

class trilinear(torch.nn.Module):

    def __init__(self, channels_in, channels_out, f_data_type=torch.float64):
        super().__init__()
        self.channels_in = channels_in
        self.channels_out = channels_out
        self.weights = torch.nn.Parameter(torch.empty((channels_out,
                                        channels_in*(channels_in + 1) // 2,
                                        channels_in), dtype=f_data_type))
        torch.nn.init.normal_(self.weights, std = 0.3)
        self.weights_i = torch.nn.Parameter(torch.empty((channels_out,
                                        channels_in*(channels_in + 1) // 2,
                                        channels_in), dtype=f_data_type))
        torch.nn.init.normal_(self.weights_i, std = 0.3)
        self.maps = []
        for i in range(channels_in):
            for j in range(i, channels_in):
                self.maps.append((i, j))

    def forward(self, phi):
        new = torch.zeros((*phi[:-3], self.channels_out, 3, 3), dtype=phi.dtype,
                          device=phi.device)
        for i in range(self.channels_in * (self.channels_in + 1) // 2):
            new += torch.einsum("ni,...ab,...icb,...cd->...nad",
                                self.weights[:,i,:], phi[...,self.maps[i][0],:,:],
                                phi.conj(), phi[..., self.maps[i][1], :, :])
            new += torch.einsum("ni,...ab,...icb,...cd->...nad",
                                self.weights[:,i,:], phi[...,self.maps[i][1],:,:],
                                phi.conj(), phi[..., self.maps[i][0], :, :])
        return new
```

Convolutional Layer

```
import torch

class convolutional(torch.nn.Module):
    def __init__(self, channels_in, channels_out, f_data_type=torch.float64):
        super().__init__()
        self.channels_in = channels_in
        self.channels_out = channels_out

        self.weights_0=torch.nn.Parameter(torch.empty((channels_out,channels_in),
                                                      dtype=f_data_type))
        torch.nn.init.normal_(self.weights_0, std = 0.3)

        self.weights_dup=torch.nn.Parameter(torch.empty((channels_out,channels_in),
                                                        dtype=f_data_type))
        torch.nn.init.normal_(self.weights_dup, std = 0.3)

        self.weights_x0up=torch.nn.Parameter(torch.empty((channels_out,channels_in)
    ,
```

```
                                                        dtype=f_data_type))
        torch.nn.init.normal_(self.weights_x0up, std = 0.3)


    def forward(self, config, phi, direc, r):
        Nx, Ny, Nz, Nt = config.shape[-7:-4]
        new = torch.einsum("nm,...mij->...nij", self.weights_0 + 0j, phi)

        x0, x1 = ((1, 2), (2, 0), (0, 1))[direc]

        # positive direc direction
        phi_shifted = torch.roll(phi, -1, dims=-7+direc)
        config_shifted = torch.roll(config, -r, dims=-7+direc)
        new_tmp = torch.einsum("...ij,...cjk,...lk->...cil", config[...,direc,:,:],
                               phi_shifted[...,:,:,:],
                               config_shifted[...,direc,:,:].conj())

        new += torch.einsum("nm,...mij->...nij", self.weights_dup + 0j, new_tmp)

        # positive x0 direction
        phi_shifted = torch.roll(phi, -1, dims=-7+x0)
        config_shifted = torch.roll(config, -r, dims=-7+direc)
        new_tmp = torch.einsum("...ij,...cjk,...lk->...cil", config[...,x0,:,:],
                               phi_shifted[...,:,:,:],
                               config_shifted[...,x0,:,:].conj())

        new += torch.einsum("nm,...mij->...nij", self.weights_x0up + 0j, new_tmp)

        # positive x1 direction
        phi_shifted = torch.roll(phi, -1, dims=-7+x1)
        new_tmp = torch.einsum("...ij,...cjk,...lk->...cil", config[...,x1,:,:],
                                phi_shifted[...,:,:,:],
                                config_shifted[...,x1,:,:].conj())

        new += torch.einsum("nm,...mij->...nij", self.weights_x0up + 0j, new_tmp)

        # negative direc direction
        phi_shifted = torch.roll(phi, 1, dims=-7+direc)
        config_shifted = torch.roll(config, 1, dims=-7+direc)
        config_shifted2 = torch.roll(config_shifted, -r, dims=-7+direc)
        new_tmp = torch.einsum("...ji,...cjk,...kl->...cil",
                               config_shifted[...,direc,:,:].conj(),
                               phi_shifted[...,:,:,:],
                               config_shifted2[...,direc,:,:])
        new += torch.einsum("nm,...mij->...nij", self.weights_dup + 0j, new_tmp)

        # negative x0 direction
        phi_shifted = torch.roll(phi, 1, dims=-7+x0)
        config_shifted = torch.roll(config, 1, dims=-7+x0)
        config_shifted2 = torch.roll(config_shifted, -r, dims=-7+direc)
        new_tmp = torch.einsum("...ji,...cjk,...kl->...cil",
                               config_shifted[...,x0,:,:].conj(),
                               phi_shifted[...,:,:,:], config_shifted2[...,x0,:,:])
        new += torch.einsum("nm,...mij->...nij", self.weights_x0up + 0j, new_tmp)

        # negative x1 direction
        phi_shifted = torch.roll(phi, 1, dims=-7+x1)
        config_shifted = torch.roll(config, 1, dims=-7+x1)
        config_shifted2 = torch.roll(config_shifted, -r, dims=-7+direc)
        new_tmp = torch.einsum("...ji,...cjk,...kl->...cil",
                               config_shifted[...,x1,:,:].conj(),
                               phi_shifted[...,:,:,:], config_shifted2[...,x1,:,:])
        new += torch.einsum("nm,...mij->...nij",self.weights_x0up+0j,new_tmp)
```

```
81
82          return new
```

Acknowledgements

I want to thank Prof. Dr. Nora Brambilla for supervising me and letting me work on such a topical project. I also want to thank Julian Mayer-Steudte for introducing me to the world of lattice QCD and machine learning and for his guidance throughout this project.