Machine learning high multiplicity matrix elements for electron-positron and hadron-hadron colliders

TRUONG, HENRY

Machine learning high multiplicity matrix elements for electron-positron and hadron-hadron colliders

Henry Truong

A Thesis presented for the degree of
Doctor of Philosophy

Institute for Particle Physics Phenomenology
Department of Physics
Durham University
United Kingdom

March 2023

Machine learning high multiplicity matrix elements for electron-positron and hadron-hadron colliders

Henry Truong

Submitted for the degree of Doctor of Philosophy

March 2023

Abstract: The LHC is a large-scale particle collider experiment collecting vast quantities of experimental data to study the fundamental particles, and forces, of nature. Theoretical predictions made with the SM can be compared with observables measured at experiments. These predictions rely on the use of Monte Carlo event generators to simulate events which demand the evaluation of a matrix element. For high multiplicity processes this can take up a significant portion of the time spent simulating an event. In this thesis, we explore the usage of machine learning to accelerate the evaluation of matrix elements by introducing a factorisation-aware neural network model. Matrix elements are plagued with singular structures in regions of phase-space where particles become soft or collinear, however, the behaviour of the matrix element in these limits is well-understood. By exploiting the factorisation property of matrix elements in these limits, the model can learn how to best represent the approximation of the matrix elements as a linear combination of singular functions. We examine the application of the model to e^-e^+ annihilation matrix elements at tree-level and one-loop level, as well as to leading order *pp* collisions where the acceleration of event generation is critical for current experiments.

Contents

List of Figures

List of Tables

Declaration

The work in this thesis is based on research carried out in the Department of Physics at Durham University. No part of this thesis has been submitted elsewhere for any degree or qualification.

Research presented in this thesis is based on joint work:

- Chapter 5 is based on [5]: Daniel Maître and Henry Truong, *A factorisation-aware Matrix element emulator*, Journal of High Energy Physics, 2021 (11). 066.

- Chapter 6 is based on research with Daniel Maître in preparation to be published.

- Chapter 7 is based on research with Timo Janßen, Daniel Maître, Steffen Schumann, and Frank Siegert in preparation to be published.

Other research projects published during my studies but not included in this thesis:

- [6]: J. Aylett-Bullock, C. Cuesta-Lazaro, A. Quera-Bofarull, M. Icaza-Lizaola, A. Sedgewick, H. Truong, A. Curran, E. Elliott, T. Caulfield, K. Fong, I. Vernon, J. Williams, R. Bower and F. Krauss, *June: open-source individual-based epidemiology simulation*, (2021), R. Soc. open sci.8210506210506.

- [7]: I. Vernon, J. Owen, J. Aylett-Bullock, C. Cuesta-Lazaro, J. Frawley, A. Quera-Bofarull, A. Sedgewick, D. Shi, H. Truong, M. Turner, J. Walker, T. Caulfield, K. Fong and F. Krauss, *Bayesian emulation and history matching of JUNE*, (2022), Phil. Trans. R. Soc. A.38020220039220220039.

Acknowledgements

The work carried out during this thesis would not have been possible without all the support from the people around me. Firstly, I would like to express my gratitude to Michael Spannowsky, and to my supervisor Daniel Maître, who gave me a chance to undertake this doctorate when nobody else did. Daniel has always been supportive with his meticulous advice and guidance, and I thank him for giving me the freedom to explore different avenues during our joint research projects.

Everyone at the IPPP, especially Trudy and Joanne, has fostered a comfortable working and social environment of which I was very glad to be a part of. To all the students who have come and gone, it was a pleasure to get to know you all. Special mentions go to Alan, Asli, Dorian, Francesco, Joe, Kevin, Marian, Ryan, and of course all the wonderful people in OC215 and beyond. I would also like to thank the proofreaders: Dan, Francesco, Guillaume, Hitham, Jack S, Joe AB, Oscar, Peter, Ryan, and Tommy. Outside of the particles, shout-out to Aidan, Cameron, Ed, Jack, and Vicky for being great friends who I could always have a laugh with and chat about anything. I want to say thank you to Jack in particular for being a friend I could always count on for a good time, and a serious time.

I want to express my deepest gratitude to my dear family and friends back home. Mum, dad, Annie, and Tracy, Chris and Tyrone, your endless love and support has been more helpful than you can imagine in completing this journey.

These past four years of my life have been tremendously formative in many ways. All the people I have met, I have learned so much from, and I will cherish all the memories that we shared together. I will leave Durham as a better person.

Chapter 1

Introduction

The Large Hadron Collider (LHC) at CERN has been colliding particles since 2009, with each successive run increasing the centre-of-mass energy of the collision, from 7 TeV in Run 1 to 13.6 TeV at the time of writing. As the experiment has operated for over a decade, there is a vast quantity of experimental data collected that has to be compared with theoretical predictions. Currently, the theory predictions are made with the Standard Model (SM) of particle physics which describes the fundamental particles of nature and their interactions.

The standard paradigm for comparing theory and experiment is to simulate the particle collisions from the initial collision all the way through to the detection of the hundreds of particles produced in the collision. General purpose Monte Carlo event generators are the de facto tool designed to generate these simulated events. With the large number of scattering events collected at the LHC already, and with even more expected at the High Luminosity LHC (HL-LHC), the ability to generate the large simulated event samples required for theory predictions within the available computing budget presents a very real challenge.

The process of generating simulated events can be broadly split up into three segments: the highly energetic hard scattering process, the subsequent cascading of particles (parton shower), and the formation of bound states detected (hadronisation), which makes their way to the detectors. While hadronisation is based on phenomenological models, the hard scattering and parton shower are derived from the SM and are perturbatively defined.

The hard scattering cross-section can be formulated as a probability for particles to collide, represented by matrix elements. These matrix elements are formally calculated order-by-order in perturbation theory, with each order leading to increasingly difficult computations at each order. In practice, these matrix elements are provided in libraries interfaced to the event generators and evaluation is largely automated for many of the most important processes. However, the evaluation time for these complex expressions becomes problematic when generating the large samples required to compare with experiments. Especially when a significant portion of the time spent simulating an event is spent on evaluating matrix elements.

The focus of this thesis will be on applying modern machine learning methods to accelerate the evaluation of matrix elements in order to speed up the event generation process. Machine learning algorithms have become ubiquitous in many fields, and their applicability to a wide range of problems have made them a popular choice in the high energy physics community as well. By combining the well understood behaviour of matrix elements in specific kinematic regions with powerful machine learning algorithms, the construction of physics-inspired machine learning models unlocks higher levels of accuracy than would otherwise be achievable.

The structure of this thesis is as follows: in this chapter I will recap the necessary concepts from the SM, in particular quantum chromodynamics (QCD), the sector which governs the behaviour of quarks and gluons. Furthermore, I will elaborate on the relationship between theory and experiment with the introduction of cross-sections, and their relation to matrix elements. In Chapter 2, I will discuss the challenges that arise in fixed-order perturbative calculations and the current methods that have been adopted in the community to deal with them. I will briefly introduce Monte Carlo event generators in Chapter 3, where the theory discussed in the first two chapters is applied in practice. In Chapter 4, I discuss how bottlenecks in traditional event generator techniques motivates the usage of machine learning based approaches, with particular attention placed on using neural networks as emulators for matrix elements. The construction of this neural network emulator is described in detail in Chapter 5 for tree-level electron-positron annihilation matrix element emulation. A similar philosophy is applied in Chapter 6 where next-to-leading order QCD k-factors are emulated for the same processes. These two chapters discuss in detail the procedure for constructing emulators for electron-positron annihilation, however, the more relevant processes for the LHC are in proton-proton initiated collisions. The extension to this scenario is detailed in Chapter 7 where I also explore using the emulator in a novel implementation to accelerate event unweighting in the event generator SHERPA. Finally, I will summarise and conclude the thesis in Chapter 8.

1.1 The Standard Model of particle physics

The Standard Model of particle physics is our current best working theory to describe all known elementary particles, as well as three of the four fundamental forces. Developed predominantly in the latter half of the 20th century, it is one of the most well tested theories that we have in science today. Some highlights include the highly precise predictions of the anomalous magnetic moment of the electron, which agrees with experimental measurements to more than 10 significant figures [8], and the discovery of the Higgs boson in 2012 by the ATLAS [9] and CMS experiments [10] at the LHC, which was theorised decades prior.

The SM is a gauge quantum field theory (QFT) where particles are described as excitations of quantum fields. The symmetry group of the SM is

$$\mathrm{SU}(3)_c \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \,, \tag{1.1.1}$$

where subscripts denote the charges of the gauge groups. The first gauge group with colour charge c describes the interactions of the strong force within the theory of quantum chromodynamics, which we will elaborate more on in Section 1.2. The second and third gauge groups represent the electroweak sector of the SM, where

L denotes left-chiral fields that carry weak isospin, whilst Y denotes the weak hypercharge. Under electroweak spontaneous symmetry breaking (EWSB), this product becomes

$$\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \xrightarrow{\mathrm{EWSB}} \mathrm{U}(1)_{\mathrm{EM}} \,, \tag{1.1.2}$$

giving rise to the electromagnetic and weak forces that we observe. The explanation of gravity in a QFT is an open problem and cannot currently be included in the SM as any Lagrangian including gravity cannot be renormalised. Fortunately, the effect of gravity is considered to be negligible on the scales considered at high energy colliders, and so it is ignored. Each of the three fundamental forces described by the SM is mediated by the exchange of a gauge boson. The massless gauge bosons mediating the strong and EM forces are the gluon g and photon γ, respectively. For the weak force the gauge bosons are the $W^{\pm}$ and Z^0 bosons, which attain a mass through the Higgs mechanism [11–13] during EWSB, elucidating the Higgs boson H.

The matter content of the SM consists of fermions, which can be split into quarks and leptons. Quarks are massive and experience the strong, weak, and EM forces. Leptons are defined by their lack of colour charge, meaning they do not experience the strong force. Leptons can be separated into charged leptons (electron e, muon μ, tau τ and their antiparticles) which experience the weak and EM force, and neutrinos (electron neutrino ν_e, muon neutrino ν_μ, tau neutrino ν_τ and their antiparticles) which only experience the weak force. Neutrinos are massless in the SM, however, they have been observed to have mass [14, 15]. This prompts physics beyond the Standard Model (BSM) to describe these observed masses.

The particle content of the SM is summarised in Figure 1.1, which shows the quarks, leptons and bosons along with their masses, charges and spin. The interactions of

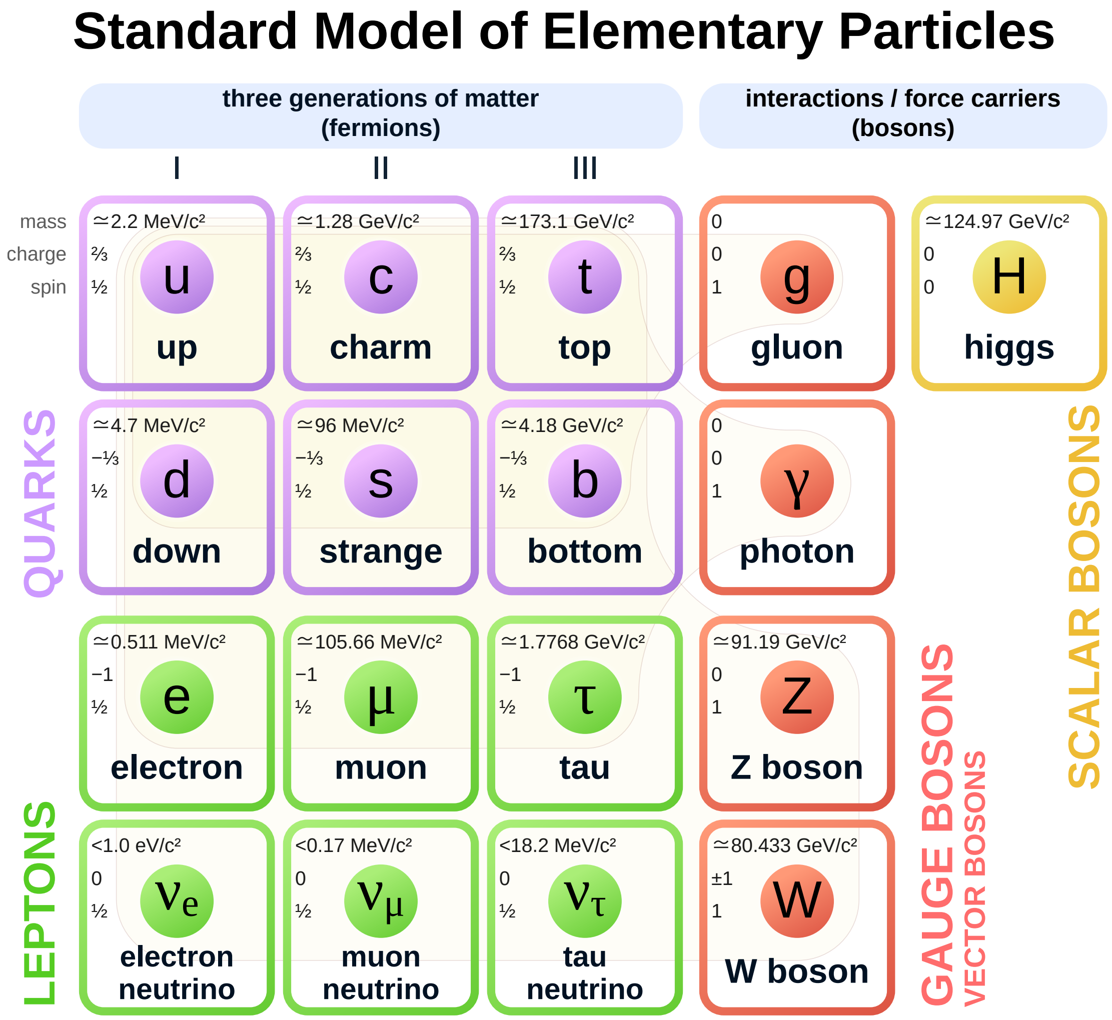


Figure 1.1: Particle content of the SM, split into quarks, leptons, gauge bosons and scalar bosons. The columns for the fermions depict the three different generations. The masses, electric charge, and spin are given for each particle. The yellow contours indicate the coupling of bosons to fermions, illustrating the forces experienced by the fermions inside the contour. Figure reproduced from [1].

these fields are governed by the SM Lagrangian[1] which can be written as

$$\mathcal{L}_{\text{SM}} = \mathcal{L}_{\text{gauge}} + \mathcal{L}_{\text{fermion}} + \mathcal{L}_{\text{Higgs}} + \mathcal{L}_{\text{Yukawa}} + \mathcal{L}_{\text{GF}} + \mathcal{L}_{\text{ghost}} \,, \tag{1.1.3}$$

where $\mathcal{L}_{\text{gauge}}$ describes the gauge fields, $\mathcal{L}_{\text{fermion}}$ describes how fermions interact with gauge fields as well as their kinetic terms, $\mathcal{L}_{\text{Higgs}}$ describes the Higgs field, $\mathcal{L}_{\text{Yukawa}}$ describes the interactions between the Higgs field and fermions, $\mathcal{L}_{\text{GF}}$ is a gauge fixing term, and $\mathcal{L}_{\text{ghost}}$ is a ghost term. The last two terms are required to remove unphysical degrees of freedom when gauge fixing the theory. All terms in the SM Lagrangian are invariant under local transformations of the gauge group in Eq. (1.1.1).

In the following section we will focus on the gauge, fermion, gauge-fixing and ghost Lagrangian terms in the framework of QCD, which is the most relevant sector for this thesis. The remaining terms are discussed at length in standard reference texts [16–18].

1.2 Introduction to quantum chromodynamics

1.2.1 The QCD Lagrangian

Quantum chromodynamics is the sector of the SM that describes the strong interaction. QCD is a non-Abelian gauge theory with gauge group SU($N_c = 3$) where the charge is named colour. The gauge and fermion part of the QCD Lagrangian is

$$\mathcal{L}_{\text{QCD}} = -\frac{1}{4} F^a_{\mu\nu} F^{a,\mu\nu} + \sum_f \bar{\psi}^f_i (\mathrm{i}\not{D}_{ij} - \delta_{ij} m_f) \psi^f_j \,, \tag{1.2.1}$$

where repeated indices are summed over. The fields ψ^f_i are the fermions field operators, representing quarks and antiquarks with flavours f: up u, down d, strange s, charm c, top t, and bottom b, with masses m_f. The field operators transform

[1] Technically Lagrangian density but we use the terms Lagrangian and Lagrangian density interchangeably.

under the fundamental representation with indices $i, j \in \{1, 2, 3\}$, named colour indices. The gauge fields A^a_μ, corresponding to gluons, appear in the quark covariant derivative

$$(D_\mu)_{ij} = \delta_{ij}\partial_\mu - \mathrm{i}g_s T^a_{ij} A^a_\mu \,, \tag{1.2.2}$$

and the gauge field strength tensor

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g_s f^{abc} A^b_\mu A^c_\nu \,. \tag{1.2.3}$$

Gauge fields transform under the adjoint representation, which is an $N_c^2 - 1$ dimensional representation such that the adjoint indices $a, b, c \in \{1, ..., 8\}$. T^a_{ij} are the group generators in the fundamental representation. In SU(3) it is common to write the group generators as

$$T^a_{ij} = \frac{1}{2}\lambda^a_{ij} \tag{1.2.4}$$

where λ^a_{ij} are the Gell-Mann matrices [19]. The generators of the group obey the Lie algebra

$$[T^a, T^b] = \mathrm{i}f^{abc}T^c \,, \tag{1.2.5}$$

where f^{abc} are the structure constants of SU(3). By convention, the generators are normalised to be

$$\mathrm{Tr}(T^a T^b) = \delta^{ab} T_R \,, \quad \text{where} \quad T_R = \frac{1}{2} \,. \tag{1.2.6}$$

This normalisation sets the values of the Casimirs of the group as

$$\begin{aligned} T^a_{ij} T^a_{jk} &= \delta_{ik} C_F \,, \quad C_F = \frac{N_c^2 - 1}{2N_c} \,, \\ f^{abc} f^{abd} &= \delta^{cd} C_A \,, \quad C_A = N_c \,, \end{aligned} \tag{1.2.7}$$

where $C_F = 4/3$ and $C_A = 3$ for QCD. These are collectively referred to as colour factors.

The gauge coupling of the group g_s is a dimensionless free parameter of the theory. It is common to use the strong coupling constant instead,

$$\alpha_s = \frac{g_s^2}{4\pi} \,. \tag{1.2.8}$$

The strong coupling constant is in fact not constant, and depends on the energy scale of the process. This is due to the process of renormalisation, which will be discussed in Section 2.1.2. The implication of this is that at collider experiments where collisions occur at extremely high energy, the strong coupling constant becomes small, a property known as asymptotic freedom. A consequence of this is that predictions made in QCD can be expressed in the form of perturbative expansions in α_s (see Section 1.4).

To remove unphysical degrees of freedom from the theory we need to fix the gauge and add a ghost Lagrangian. The gauge-fixing term in the R_ξ gauge is written as

$$\mathcal{L}_{\mathrm{GF}} = -\frac{1}{2\xi}(\partial^\mu A^a_\mu)^2\,, \tag{1.2.9}$$

where $\xi = 1$ corresponds to the Feynman-'t Hooft gauge. Ghosts and antighosts are anti-commuting fields introduced for each gauge field as a way to conveniently compute determinants occurring in the gauge fixing procedure. The ghost Lagrangian is most commonly written in the Faddeev-Popov procedure as

$$\mathcal{L}_{\mathrm{ghost}} = (\partial_\mu \bar{c}^a)(\delta^{ac}\partial_\mu + g_s f^{abc} A^b_\mu)c^c\,, \tag{1.2.10}$$

where c^a ($\bar{c}^a$) are Faddeev-Popov ghosts (antighosts). Ghosts are particles that subtract the longitudinal degree of freedom from gluons, and as such it is usual to use them to cancel the longitudinal degree of freedom of gluon propagators. It should be noted that ghosts could in principle be included as external states as well for the same purpose, however, it is more conventional to use the usual physical polarisation sum instead.

1.2.2 QCD Feynman rules

Now that we have written down the QCD Lagrangian, we can examine the terms that govern the interactions between bosons and fermions, and especially important for QCD, the gauge boson self interactions. A convenient way to do this is by deriving the Feynman rules of the theory, which can be pieced together to form Feynman diagrams. We will see that Feynman diagrams are a simple but powerful tool to systematically build up all the possible ways in which a process can occur.

Expanding Eq. (1.2.1) and extracting the terms that mix the gauge and fermion fields, we get an interaction Lagrangian

$$\mathcal{L}_{\text{int}} = g_s A^a_\mu \bar{\psi}^f_i \gamma^\mu T^a_{ij} \psi^f_j - g_s f^{abc}(\partial_\mu A^a_\nu) A^{b,\mu} A^{c,\nu} - \frac{g_s^2}{4} f^{abc} A^b_\mu A^c_\nu f^{ade} A^{d,\mu} A^{e,\nu} \,, \tag{1.2.11}$$

which we can interpret as follows: the first term is an interaction between a gluon and two fermion field operators, the second term is a three gluon self-interaction, and the third term is a four gluon self-interaction. These mixing terms can be recast into Feynman rules representing interaction vertices.

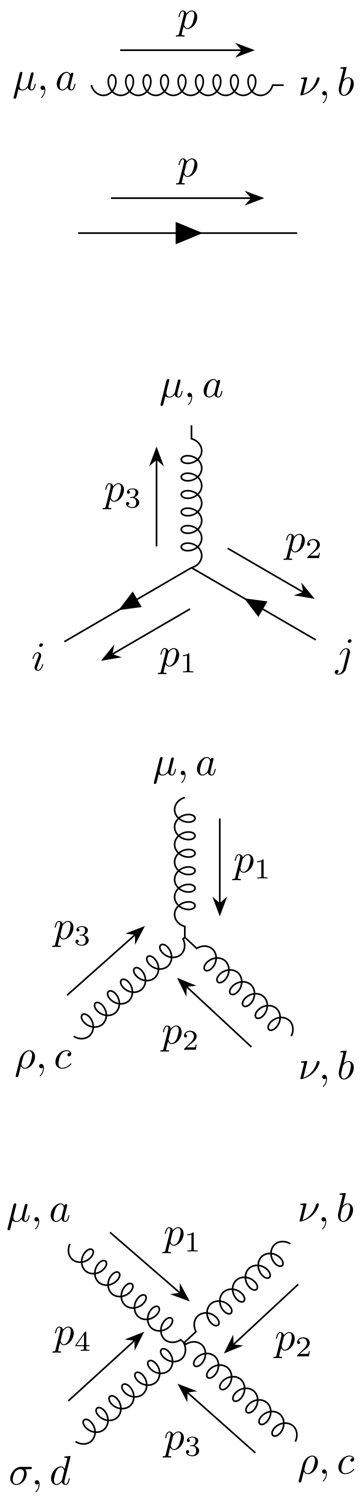
To connect vertices we require propagators, which we can read off from the kinetic Lagrangian where we collect terms from Eqs. (1.2.1) and (1.2.9),

$$\mathcal{L}_{\text{kin}} = -\frac{1}{4}(\partial_\mu A^a_\nu - \partial_\nu A^a_\mu)^2 - \frac{1}{2\xi}(\partial_\mu A^a_\mu)^2 + \bar{\psi}^f_i (\mathrm{i}\not{\partial} - m_f) \psi^f_i \,, \tag{1.2.12}$$

where the first two terms corresponds to a gluon propagator and the third term corresponds to a quark propagator.

From these Lagrangians, we can read off the QCD Feynman rules which we have collected in Figure 1.2. Note that we have only written down the vertices and propagators for the physical gluons and quarks. Ghosts also have associated Feynman rules but we do not specify those here. A full list of Feynman rules in the SM can be found in Ref. [18].

From these Feynman rules, we identify that each fermion-gluon and three gluon vertex is associated with one power of α_s (due to squaring g_s), and the four gluon vertex is associated with two powers of α_s. This relationship between vertices and α_s allows us to systematically build up Feynman diagrams that correspond to a fixed-order in α_s. This point will be elaborated on in Section 1.4 where we discuss scattering amplitudes and how they are related to Feynman diagrams.



$$= -\mathrm{i}\delta_{ab}\left[\frac{g^{\mu\nu}}{p^2+i\epsilon} - (1-\xi)\frac{p^\mu p^\nu}{(p^2)^2}\right]$$

$$= \frac{\mathrm{i}(\not{p}+m_f)}{p^2-m_f^2+i\epsilon}$$

$$= -\mathrm{i}g_s\gamma^\mu T^a_{ij}$$

$$\begin{aligned} = -g_s f^{abc}\big[&g^{\mu\nu}(p_1-p_2)^\rho \\ &+g^{\nu\rho}(p_2-p_3)^\mu \\ &+g^{\rho\mu}(p_3-p_1)^\nu\big] \end{aligned}$$

$$\begin{aligned} = -\mathrm{i}g_s^2\big[&f^{eab}f^{ecd}(g^{\mu\rho}g^{\nu\sigma}-g^{\mu\sigma}g^{\nu\rho}) \\ &+f^{eac}f^{edb}(g^{\mu\sigma}g^{\rho\nu}-g^{\mu\nu}g^{\rho\sigma}) \\ &+f^{ead}f^{ebc}(g^{\mu\nu}g^{\rho\sigma}-g^{\mu\rho}g^{\nu\sigma})\big] \end{aligned}$$

Figure 1.2: QCD Feynman rules with the Minkowski metric $g^{\mu\nu} = \mathrm{diag}(1, -1, -1, -1)$.

1.3 Factorisation theorem

At the LHC collisions occur between two protons, which have constituent quarks and gluons, collectively named partons. The application of QCD to describe phenomena in these proton-proton collisions rests on the use of the factorisation theorem, which enables the separation of low and high energy scales (so-called soft and hard). Within the proton, there are quarks-antiquark pairs and gluons that are constantly absorbed and emitted on a timescale inversely proportional to the mass of the proton. The timescale of this fluctuation is much longer than the timescale of the interaction between a parton and a highly energetic probing parton from another proton. This is the scenario that occurs in collisions at the LHC. During the collision, or the hard scattering, the probe is able to interact with a parton that is effectively frozen. The probing parton knows nothing of the proton being probed, except for the fact that it collided with a parton carrying a fraction of the proton momentum. The distribution of partons within a proton is process independent and a fundamental property of the proton, meaning it can be separated from the actual scattering between the partons.

This heuristic argument motivates the form of the factorisation equation, where the cross-section, which is proportional to the production rate of particles from a hadronic collision, can be written as

$$\sigma_{AB\to n} = \sum_{a,b} \int_0^1 \mathrm{d}x_a \mathrm{d}x_b \; f_{a/A}(x_a, \mu_F) f_{b/B}(x_b, \mu_F) \; \hat{\sigma}_{ab\to n}(Q, \mu_F, \mu_R) + \mathcal{O}\left(\frac{\Lambda_{\mathrm{QCD}}}{Q}\right) , \tag{1.3.1}$$

where a is the parton from hadron A, and b is the parton from hadron B. The sum over initial-state partons a and b indicates the sum over flavours, which depends on the hadron composition in general. This factorisation is not exact as indicated by the correction term, which is inversely proportional to the characteristic hard scale Q. The other scale involved, Λ_{QCD}, the QCD scale, is the scale at which α_s becomes large enough that perturbation theory breaks down. However, for high energy collisions, where $Q \gg \Lambda_{\mathrm{QCD}}$, this term and higher order terms are suppressed. In fact, the factorisation equation has only been proven for a few specific cases [20–23].

The parton distribution functions (PDFs), $f_{a/h}(x,\mu)$, depend on the momenta fraction x carried by parton a, with respect to its parent hadron h, at a scale μ, usually taken to be the factorisation scale μ_F. The factorisation scale is the interface between soft and hard physics. The interpretation of PDFs at leading order are as probability distributions: $f_{a/h}(x,\mu)$ is the probability of finding parton a within h carrying a momentum fraction x at the energy scale μ. PDFs are non-perturbative objects that encapsulate the soft effects of the scattering occurring below energy μ_F. They are non-perturbative because they are determined by fitting to experimental data, instead of being calculated in perturbation theory.

For a review of PDFs and how they are determined see Refs. [24,25]. In practice, these PDF fits are accessed through the LHAPDF interface [26] which provides PDF sets from multiple working groups [27–30]. The evolution of PDFs between factorisation scales is possible through the use of the Dokshitzer-Gribov-Lipatov-Altarelli-Parisi (DGLAP) equations [31–34],

$$\mu_F^2 \frac{\partial}{\partial \mu_F^2} \begin{pmatrix} f_{q/h}(x,\mu_F) \\ f_{g/h}(x,\mu_F) \end{pmatrix} = \frac{\alpha_s(\mu_F)}{2\pi} \int_x^1 \frac{\mathrm{d}z}{z} \begin{pmatrix} P_{qq}(\frac{x}{z}) & P_{qg}(\frac{x}{z}) \\ P_{gq}(\frac{x}{z}) & P_{gg}(\frac{x}{z}) \end{pmatrix} \begin{pmatrix} f_{q/h}(z,\mu_F) \\ f_{g/h}(z,\mu_F) \end{pmatrix}, \quad (1.3.2)$$

where $P_{ab}(\frac{x}{z})$ are splitting functions representing parton b emitting a parton a, that carries a momentum fraction x. These splitting functions are calculable as a power series in α_s where they have been computed up to three-loops [35, 36]. At leading order they are [34]

$$\begin{aligned}
P_{qq}^{(0)}(z) &= C_F \left[\frac{1+z^2}{(1-z)_+} + \frac{3}{2}\delta(1-z)\right], \\
P_{qg}^{(0)}(z) &= T_R \left[z^2 + (1-z)^2\right], \\
P_{gq}^{(0)}(z) &= C_F \left[\frac{1+(1-z)^2}{z^2}\right], \\
P_{gg}^{(0)}(z) &= 2C_A \left[\frac{z}{(1-z)_+} + \frac{1-z}{z} + z(1-z)\right] + \left(\frac{11C_A - 4T_R n_f}{6}\right)\delta(1-z),
\end{aligned} \quad (1.3.3)$$

where the divergences at $z = 1$ have been regulated with the '+'-prescription, which is defined as

$$\int_0^1 \mathrm{d}z\, [g(z)]_+ f(z) = \int_0^1 \mathrm{d}z\, g(z)(f(z) - f(1))\,. \tag{1.3.4}$$

With this prescription, the divergence at $g(z = 1)$ is cancelled, given that the function $f(z)$ is sufficiently smooth at $z = 1$. At leading order, there is a physical interpretation of the splitting kernels as the probability of finding a parton a in parton b carrying a momentum fraction x of the parent parton.

The momentum fractions, x, also link the squared centre-of-mass energies of the hadronic collision, denoted as s, to the partonic equivalent as

$$\hat{s} = x_a x_b s\,. \tag{1.3.5}$$

The partonic cross-section, $\hat{\sigma}_{ab\to n}(Q, \mu_F, \mu_R)$, describes the interaction of partons a and b scattering into n particles, where the scattering occurs at energy scale Q, which is often taken to be $\hat{s}$. Notice that the partonic cross-section depends on both the factorisation scale, μ_F, and the renormalisation scale, μ_R (see Section 2.1.2). This dependence will be discussed in Section 3.4 in the context of theoretical uncertainties.

The discussion so far has been focused on hadron-hadron initiated scattering, however a similar argument can be made about electron-positron annihilation, where the electron-positron annihilates to form a photon or Z boson which decays into a quark-antiquark pair. For a sufficiently high energy collision, these interactions can be calculated in perturbation theory. Although electrons and positrons are fundamental particles and not composite particles, the cross-sections in electron-positron collisions have contributions from the initial-state radiation, which also reduces the energy of the hard scattering. One method to capture these effects is through the use of process-independent structure functions [37] which are analogous to PDFs. Therefore, the discussion presented in this section applies to electron-positron cross-sections as well.

We have seen that cross-section computations involve the convolution of PDFs with partonic cross-sections, $\hat{\sigma}_{ab\to n}$, which are calculated in perturbation theory. The details of how partonic cross-sections are computed is discussed in the next section.

1.4 Scattering amplitudes and cross-sections

At collider experiments we typically collide two beams consisting of bunches of energetic particles and analyse the resulting products of these collisions. The predictions that we make from our theoretical model are the partonic cross-sections. To arrive at an expression for the partonic cross-section, we need to first consider the hard scattering of particles. To model the hard collisions in a collider experiment, we look at the specific case of two particles colliding and producing n particles. Scattering amplitudes are used to mathematically describe these scattering processes. For an initial-state $|i\rangle$ and final-state $|f\rangle$ the scattering amplitude can be written as the overlap between the states

$$\langle f|S|i\rangle \,, \tag{1.4.1}$$

where the scattering matrix, or S-matrix can be decomposed into an identity matrix and a transfer matrix $\mathcal{T}$,

$$S = \mathbb{1} + \mathrm{i}\mathcal{T} \,. \tag{1.4.2}$$

The S-matrix encodes all the information about how the initial-state will evolve over time. By writing the S-matrix in this way, all the interactions are separated into the transfer matrix $\mathcal{T}$, as the identity matrix describes the free theory. By imposing a momentum conservation constraint on the S-matrix, the matrix element, $\mathcal{M}$, is defined in the following expression

$$\langle f|\mathcal{T}|i\rangle = (2\pi)^4\delta^4\left(p_a + p_b - \sum_{f=1}^{n} p_f\right)\mathcal{M} \,, \tag{1.4.3}$$

where we take p_a and p_b to be the momenta of the two colliding particles in the initial-state, and p_f to be the momenta of the n particles in the final-state. The probability of this process occurring is the modulus squared of the scattering amplitude[2]

$$P = |\langle f|S|i\rangle|^2 \propto |\langle f|\mathcal{M}|i\rangle|^2 \tag{1.4.4}$$

where $|\langle f|\mathcal{M}|i\rangle|^2 \equiv |\mathcal{M}|^2$ is the matrix element squared[3].

An observable that can be measured at experiments is the cross-section, σ, as introduced in Eq. (1.3.1). The cross-section is a property of the particles being scattered, and is independent of the way the experiment is carried out (disregarding energy scale of experiment). Since the hadronic cross-section factorises the low energy effects into the non-perturbative PDFs, which are determined once, for all processes, the object of interest now becomes the partonic cross-section, $\hat{\sigma}$, which encodes the scattering information of the specific process considered.

Given that cross-sections are measurable quantities, the partonic cross-section directly relates the matrix elements which we calculate in our QFT to measurable observables at experiments. In practice, it is more useful to consider the differential cross-section, $\mathrm{d}\hat{\sigma}$, where the cross-section can be differential in quantities such as energies and angles. This is because individual events at experiments will be measured to be in a specific interval of the differential quantity, meaning differential cross-sections can be plotted as histograms. Predictions can then be made on the theory side by binning simulated events.

The differential partonic cross-section can be written as

$$\mathrm{d}\hat{\sigma} = \frac{1}{\mathcal{F}}|\mathcal{M}_{ab\to n}|^2\mathrm{d}\Phi_n\,, \tag{1.4.5}$$

where the flux factor, $\mathcal{F}$, is given by

$$\mathcal{F} = 4\sqrt{(p_a p_b)^2 - m_a^2 m_b^2}\,. \tag{1.4.6}$$

In the massless limit the flux factor reduces reduces to

$$\mathcal{F} = 2(p_a + p_b)^2 = 2\hat{s}\,. \tag{1.4.7}$$

[2] Neglecting the case of scattering with no interactions taking place.

[3] Henceforth we refer to the matrix element squared as matrix element, unless stated otherwise.

It is common to use the massless limit because the centre-of-mass energy of a collider experiment is much greater than the mass of the colliding particles, and so we can neglect the masses.

The Lorentz-invariant phase-space, $\mathrm{d}\Phi_n$, contains all the possible configurations of the n-particle final-state. Absorbing the momentum conserving δ-function from Eq. (1.4.3), we can write it as

$$\mathrm{d}\Phi_n = (2\pi)^4\delta^4\left(p_a + p_b - \sum_{f=1}^{n} p_f\right)\prod_{f=1}^{n}\frac{\mathrm{d}^4 p_f}{(2\pi)^3}\delta(p_f^2 - m_f^2)\theta(E_f)\,, \tag{1.4.8}$$

where the second δ-function restricts the final-state particles to be on mass-shell (on-shell), and the step function selects only the positive energy solution. This gives an intuitive picture of the partonic cross-section as the probability of a $2 \to n$ process occurring, summed over all the possible valid final-state configurations.

From the definition of the transition probability Eq. (1.4.4), it becomes clear that the matrix element is the object which relates back to the Lagrangian of the theory as it encodes the interactions of the particles. We saw in Section 1.2.2 that interactions could be codified into Feynman rules which are pieced together to construct Feynman diagrams. In this picture, matrix elements are exactly the sum over all Feynman diagrams[4] for a particular process. However, for a given process the exact matrix element is a sum over infinitely many Feynman diagrams. Fortunately, because α_s is small in high energy collisions, we can expand the matrix element as a perturbative series in α_s to write

$$|\mathcal{M}|^2 = \alpha_s^m|\mathcal{M}|^2_{\mathrm{LO}} + \alpha_s^{m+1}|\mathcal{M}|^2_{\mathrm{NLO}} + \alpha_s^{m+2}|\mathcal{M}|^2_{\mathrm{NNLO}} + \mathcal{O}(\alpha_s^{m+3})\,, \tag{1.4.9}$$

where m corresponds to the powers of α_s in the simplest Feynman diagrams for the process of interest. Since each term of this expansion is at a fixed-order in α_s, it is possible to systematically compute the diagrams which contribute at the given order of α_s. The set of diagrams that contribute at the lowest order in α_s are $|\mathcal{M}|^2_{\mathrm{LO}}$, where LO stands for leading order. The next term in the expansion corresponds to the

[4] Since we are working on the level of $|\mathcal{M}|^2$, this would be the sum of all interference terms.

leading order diagrams with an additional loop, or external leg, which contributes an extra factor of α_s. This term is dubbed NLO for next-to-leading order in α_s, and the following term next-to-next-to-leading order has again an additional loop or leg. In general, we have terms N^{k}LO where each additional power of α_s increases complexity of the computations, however, because α_s is small, each additional term should contribute less and less. In principle, this means that the first terms dominate the expansion, and it should be sufficient to terminate the series after a few terms to reach an acceptable level of accuracy.

In summary, the partonic cross-section of a collision reduces to the computation of the matrix elements up to a fixed-order in the coupling parameter, which is then integrated over the valid phase-space of final-state particle configurations. For hadronic collisions the partonic cross-section also needs to be convolved with the PDFs to obtain the hadronic cross-sections.

With the introduction of matrix elements and cross-sections complete, we will move the discussion onto more practical aspects of computations within the QCD framework. More specifically, we will discuss the divergent structure of matrix elements and the machinery developed to tackle these unphysical singularities.

Chapter 2

Quantum chromodynamics in practice

In the previous chapter we introduced the basic concepts of QCD and setup the framework of calculating cross-sections. Cross-sections are the most relevant quantities to compute as they can be measured experimentally, and are a property of the particles being collided, rather than depending on the specifics of the experimental procedure. This provides a bridge to compare theoretical predictions and experimental measurements. The relevant quantity calculated in perturbation theory is the partonic cross-section which is the matrix element integrated over the relevant final-state phase-space, normalised by a flux factor. The matrix elements are calculated order-by-order in the strong coupling α_s (c.f. Eq. 1.4.9), where each order corresponds to a set of Feynman diagrams that have the appropriate number of vertices, as each QCD vertex brings along a factor of α_s.

In this chapter we will discuss the challenges that arise when evaluating these Feynman diagrams and outline some solutions that have been widely adopted to circumvent these issues in order to provide real-world applicable predictions. We will consider the divergent nature of matrix elements and introduce the most widely used techniques to tame these singularities. This chapter forms the theoretical foundations upon which this thesis is built on.

2.1 Divergent structures

The computation of matrix elements essentially reduces to evaluating Feynman diagrams which are analytical expressions built up from the Feynman rules of a theory (c.f. QCD Feynman rules in Figure 1.2). In evaluating certain topologies of Feynman diagrams, integral expressions containing unconstrained momenta will give rise to singularities. The divergences associated with high energy modes are called ultraviolet (UV) divergences, and on the opposite end of the energy spectrum, there are infrared (IR) divergences associated with low energy modes.

The de facto methods to alleviate these divergences are through renormalisation for UV divergences, and through subtraction schemes for IR divergences. Both of these methods will be discussed in this chapter.

2.1.1 Ultraviolet divergences

During intermediate steps of calculations, such as the computation of loop diagrams seen in Figure 2.1, we have to evaluate integrals of the form

$$I_{\mathrm{UV}} = \int_0^\Lambda \frac{\mathrm{d}^4\ell}{(2\pi)^4} \frac{1}{\ell^2(\ell+p)^2} \sim \log\Lambda\,, \tag{2.1.1}$$

where a cut-off scale Λ has been introduced to capture the divergence as the loop momenta $\ell \to \infty$. It is clear that the integral diverges in this high energy limit, hence the name ultraviolet divergence. These divergences can be systematically removed by replacing the bare masses and fields of the theory with their measured physical values, a process named renormalisation. In practice, this amounts to introducing counterterms that exactly cancel the corresponding UV divergences at the same order in perturbation theory.

Before we introduce these counterterms, we first consider the procedure of regularisation which makes these infinities explicitly manifest. The most widely adopted method of regularisation is dimensional regularisation (DR) [38] which is based on

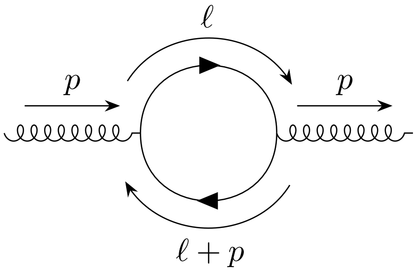


Figure 2.1: Massless bubble diagram which is an example of where unconstrained loop momenta can lead to UV divergences.

the observation that the integral in Eq. (2.1.1), which is carried out in $d = 4$ space-time dimensions, would be finite if we move away from $d = 4$ dimensions[5].

In DR we define $d = 4 - 2\epsilon$ such that after Feynman parametrisation Eq. (2.1.1) becomes

$$I_{\mathrm{UV}} = \int \frac{\mathrm{d}^d \ell}{(2\pi)^d} \frac{1}{\ell^2 (\ell + p)^2} = \frac{\mathrm{i}}{(4\pi)^2} \left(\frac{-p^2}{4\pi} \right)^{-\epsilon} \frac{\Gamma(1-\epsilon)^2}{\Gamma(2-2\epsilon)} \Gamma(\epsilon) \,, \qquad (2.1.2)$$

where the divergence for $d = 4$, or equivalently $\epsilon \to 0$, is now captured in the Gamma function, $\Gamma(\epsilon)$. This becomes clear once we expand $\Gamma(\epsilon)$ around small ϵ

$$\Gamma(\epsilon) = \frac{1}{\epsilon} - \gamma_E + \frac{1}{2} \left(\gamma_E^2 + \frac{\pi^2}{6} \right) \epsilon + \mathcal{O}(\epsilon^2) \,, \qquad (2.1.3)$$

where $\gamma_E \approx 0.577$ is the Euler-Mascheroni constant, meaning the divergence is now regularised as a pole in ϵ.

2.1.2 Renormalisation

With the UV divergence regularised by dimensional regularisation, it is now possible to construct the counterterms order-by-order in α_s to explicitly cancel the divergences.

A subtlety with adjusting the dimension of the theory from $d = 4$ to arbitrary dimensions is that the mass dimension of the Lagrangian has to change accordingly to

[5] In dimensional regularisation it is possible to regulate the integral regardless of if $d > 4$ or $d < 4$, it would amount to a sign difference.

retain a dimensionless action in natural units. We can account for this by introducing an arbitrary energy scale μ_R, the renormalisation scale, into the gauge coupling constant to modify the mass dimension of the Lagrangian. For QCD this is done via the modification

$$g_s \to g_s \mu_R^{\epsilon} \,. \tag{2.1.4}$$

where it is understood that the original gauge coupling is dimensionless. The order of μ_R is determined by examining the dimensions of the gauge and fermion fields in the Lagrangian in Eq. (1.2.1). We see that this introduces a scale dependence on the coupling constant α_s, a point we will return to in Section 2.1.3.

The inclusion of counterterms into the theory can be thought of as redefinitions of the bare fields and couplings in the QCD Lagrangian to restore predictive power to the theory. The renormalised fields and couplings can be given as (see for instance Ref. [39])

$$\begin{aligned} \psi_{\text{bare}} &= \sqrt{Z_2}\psi_R \,, \\ A^{\mu}_{\text{bare}} &= \sqrt{Z_3}A^{\mu}_R \,, \\ g_{s,\text{bare}} &= Z_g \mu_R^{\epsilon} g_{s,R} \,, \end{aligned} \tag{2.1.5}$$

where it is conventional to define $Z_1 = Z_g Z_2 \sqrt{Z_3}$ such that we can set $Z_n = 1 + \delta_n$ for $n \in \{1, 2, 3\}$. In this way we can write the renormalised Lagrangian as

$$\mathcal{L}_{\text{renorm}} = \mathcal{L}_{\text{bare}} + \mathcal{L}_{\text{c.t.}} \,, \tag{2.1.6}$$

where the counterterms appearing in $\mathcal{L}_{\text{c.t.}}$ are determined by calculating δ_n. There is freedom in the choice of the finite part of δ_n which gives rise to different regularisation schemes. The minimal subtraction (MS) scheme subtracts only the epsilon pole appearing in loop integrals. However, the most commonly used regularisation scheme is the modified minimal subtraction ($\overline{\text{MS}}$) scheme which subtracts the epsilon pole along with a universal constant appearing in all loop integrals.

In an all-orders calculation of an observable, there would be no dependence on the renormalisation scale as it is a remnant of the regularisation prescription. However, because in perturbative QCD observables are calculated order-by-order, there will be a residual dependence on the renormalisation scale stemming from the missing higher order terms. The conventional way that theoretical uncertainties associated with this residual dependence are determined is by carrying out a scale variation (see Section 3.4).

2.1.3 Running of the coupling constant

The renormalisation scale introduced during regularisation is a mathematical artifact and should not impact any measurable quantities. Therefore, the bare coupling should not depend on the renormalisation scale,

$$\frac{\mathrm{d}g_{s,\text{bare}}}{\mathrm{d}\mu_R} = 0\,, \tag{2.1.7}$$

or said another way, the renormalisation process is independent of the actual value of the renormalisation scale. The consequence of this is that the renormalised coupling, $\alpha_s(\mu_R)$, has to depend on the renormalisation scale instead.

The renormalisation scale dependence of α_s is governed by the Callan-Symanzik β-function [40, 41]

$$\mu_R^2 \frac{\partial \alpha_s(\mu_R^2)}{\partial \mu_R^2} = \beta(\alpha_s)\,, \tag{2.1.8}$$

where the β-function can be written as a perturbative expansion in α_s

$$-\beta(\alpha_s) = \alpha_s \sum_{n=0}^{\infty} \left(\frac{\alpha_s}{4\pi}\right)^{n+1} \beta_n \tag{2.1.9}$$

where the coefficients β_n have been computed up to β_4 [42, 43]. The solution to Eq. (2.1.8) to first order is

$$\alpha_s(\mu_R^2) = \frac{1}{\frac{\beta_0}{4\pi}\log\left(\frac{\mu_R^2}{\Lambda_{\text{QCD}}^2}\right)}\,, \tag{2.1.10}$$

where $\beta_0 = (11C_A - 4T_R n_f)/3$. In QCD, $C_A = 3$ and $T_R = \frac{1}{2}$, meaning for $n_f < \frac{33}{2}$ the sign of β_0 is positive. In nature we have observed six flavours of quarks, so $\beta_0 > 0$. In fact all β-coefficients computed to date are positive, meaning that α_s

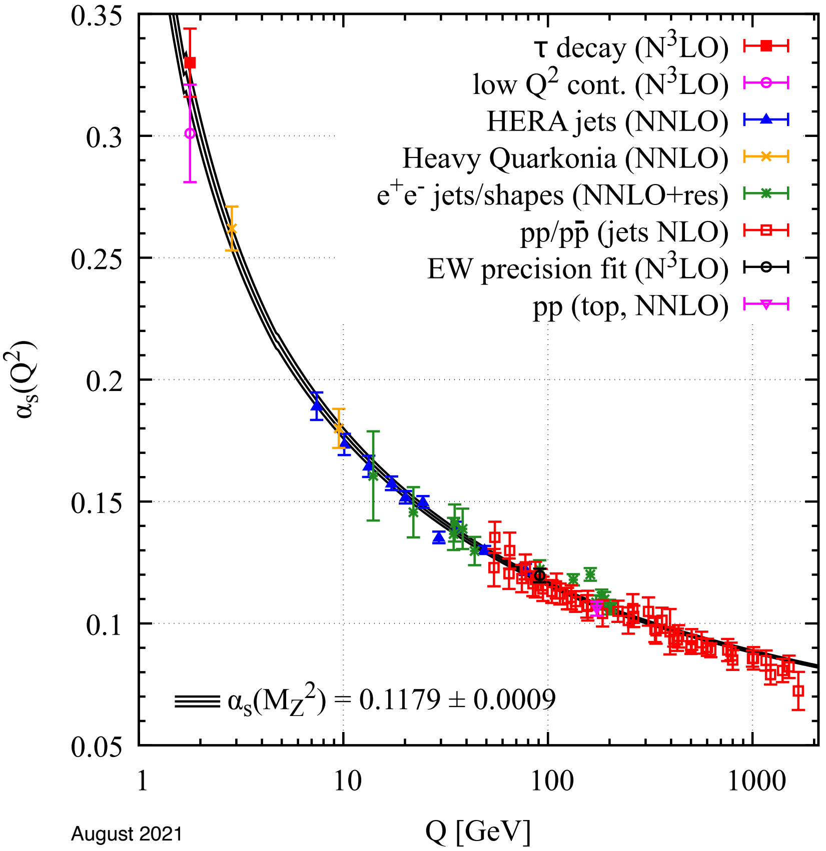


Figure 2.2: The running of the strong coupling constant α_s, as determined by experiments, with QCD theory prediction in black. Figure reproduced from [2].

decreases with increasing energy due to the minus sign in Eq. (2.1.9). This property is known as asymptotic freedom [44, 45], and is the justification for treating QCD perturbatively when the energy scale is high, such as at collider experiments. The running of the coupling constant has been observed experimentally as illustrated in Figure 2.2.

At low energies, quarks and gluons cannot exist as free particles and are forced to form composite, colourless particles. This phenomenon, named confinement, is compatible with the observation of the running coupling, but the perturbative calculation predicting the running coupling does not necessarily lead to the non-perturbative phenomenon of confinement. This means that at low energies, perturbative QCD is not an accurate description of nature, and the behaviour of hadrons cannot be predicted from perturbation theory. Instead phenomenological models have to be used to describe these processes. This will be briefly touched on in Section 3.3.

2.1.4 Infrared divergences

In Section 2.1.1 we saw how divergences stemming from high energy behaviour would occur in loop diagrams. On the other end of the energy spectrum we also have divergences from low energy modes. Since these divergences emerge from low energy behaviour they are called infrared divergences. There are two cases in which these divergences arise:

- virtual divergences in loop integrals,
- real-emission divergences when an emission of an extra particle has vanishing energy (soft), or becomes parallel to an external leg (collinear).

Virtual IR divergences can be understood by looking at an integral of the form

$$I_{\mathrm{V}} = \int \frac{\mathrm{d}^4\ell}{(2\pi)^4} \frac{1}{\ell^2(\ell+p_1)^2(\ell-p_2)^2}\,, \tag{2.1.11}$$

which is encountered when calculating the one-loop virtual correction to the gluon-fermion vertex as shown in Figure 2.3. It is clear that the denominator vanishes when $\ell \to 0$ or when either $(\ell+p_1)^2$ or $(\ell-p_2)^2 \to 0$. These situations correspond to the gluon in the loop propagator going soft, or collinear to the external quarks, respectively. These divergences are regulated in DR to give

$$I_{\mathrm{V}} = \frac{\mathrm{i}}{(4\pi)^2 Q^2}\left(\frac{-Q^2}{4\pi}\right)^{-\epsilon} \frac{\Gamma(1+\epsilon)\Gamma(1-\epsilon)^2}{\Gamma(1-2\epsilon)}\left[\frac{1}{\epsilon^2}\right]\,, \tag{2.1.12}$$

where $Q = (p_1+p_2)$. In this expression, there is a double ϵ pole manifest, corresponding to the associated soft and collinear divergences in real emission corrections.

Real IR divergences arise when integrating matrix elements over the phase-space of external state momenta. Matrix elements can be written as functions of Mandelstam variables

$$s_{ij} = (p_i+p_j)^2 \xrightarrow[\text{limit}]{\text{massless}} 2E_iE_j(1-\cos\theta_{ij})\,, \tag{2.1.13}$$

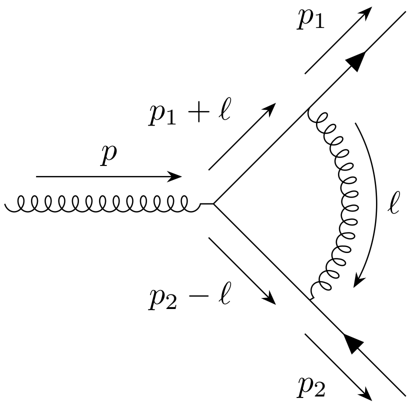


Figure 2.3: One-loop vertex correction for the gluon-quark-antiquark vertex which diverges for $\ell \to 0$.

where p_i and p_j are the 4-momenta of two partons i and j in the hard scattering. When either of the partons become soft, $E_{i,j} \to 0$, or when they go collinear to each other, $\theta_{ij} \to 0$, the matrix element will diverge if s_{ij} appears in the denominator. These scenarios correspond to an external particle becoming unresolved.

In Section 2.1.2, we removed UV divergences through the use of counterterms to renormalise the theory, making predictions from the theory physical. IR divergences, on the other hand, arise even after the theory has been renormalised. The resolution to IR divergences is given by the Block-Nordsieck [46, 47] and Kinoshita-Lee-Nauenberg (KLN) theorems [48, 49] which states that for sufficiently inclusive quantities, at each order of perturbation theory, the divergent parts of the real and virtual contributions exactly cancel out, leaving only finite corrections. This cancellation occurs because the infrared pole structure is identical in the real and virtual corrections, but with an opposite sign.

At a detector of a collider experiment, there is a finite energy resolution of the calorimeter. This means that it is not physically possible to observe arbitrarily soft particles. It is also not possible to distinguish two particles at arbitrarily small angles from one particle with the combined momenta. These limitations along with the KLN theorem means that any physical observable that we wish to predict with fixed-order perturbation theory must be insensitive to the emissions of soft or collinear particles. Any observable obeying this criteria is an infrared and collinear (IRC) safe observable. This requirement of being insensitive to additional emissions or arbitrarily soft emissions must also be met by jet definitions. Jet definitions allow theorists and experimentalists to systematically combine partons, hadrons, or energy deposits, into collimated clusters called jets. Jets provide a way to relate the produced hadrons back to the partons produced in the hard scattering as they are broadly collimated in a similar direction. For a review on jets see Ref. [50].

In DR, the singularities from soft and collinear divergences are manifest as ϵ poles which makes the cancellation simple between the real and virtual parts. However, in practice, the phase-space integrals of the matrix elements are rarely carried out analytically due to the high dimensionality of the integral, rendering them intractable. Instead they are done numerically through the use of Monte Carlo methods (see Chapter 3), which requires the integral to be in integer dimensions. Therefore, techniques have been developed to deal with these IR divergences as well. The most common technique, subtraction, will be discussed in more detail in Section 2.3.

2.2 Factorisation of matrix elements

In the previous section we examined the IR divergences that can arise in matrix elements, either explicitly through loop diagrams, or implicitly when carrying out the phase-space integral over the external state momenta. In this section we will discuss how these IR singularities are universal and how the divergences associated with real emissions can be factorised out of the matrix element. This property is exploited extensively in Chapters 5, 6, and 7 where the research of this thesis is presented.

The regions of phase-space in which real emission matrix elements diverge are when the emission is soft and/or collinear. It can be shown that in these regions of phase-space, the matrix element factorises into a process-independent singular factor, multiplied by a matrix element with the unresolved parton being absorbed by an emitting leg – we call this the reduced matrix element. This factorisation, however, is not exact in QCD where there are spin and colour correlations.

Schematically, the $(n+1)$-body matrix element factorises as

$$|\mathcal{M}_{n+1}|^2 \to \boldsymbol{S}_{ijk} \otimes |\mathcal{M}_n|^2 \,, \tag{2.2.1}$$

where $\boldsymbol{S}_{ijk}$ is a universal singular factor capturing the IR divergent behaviour and $|\mathcal{M}_n|^2$ is the reduced matrix element. The $\otimes$ represents the colour and spin correlations existing between the singular function and the reduced matrix element. The indices of $\boldsymbol{S}_{ijk}$ already hints at the fact that the singular functions only depends on three partons, and not all of the external states. This singular factor is not unique and can be represented by different approximations as long as they reproduce the correct IR behaviour. In this thesis we discuss two approximations: Catani-Seymour dipoles [51] in Section 2.4 and antenna functions [52] in Section 2.5. The indices ijk will be given specific assignments in these two different approximations.

We will inspect the soft and collinear limits separately to see how the matrix element factorises in these respective limits. The following section will follow the conventions of Ref. [51], namely that the colour and helicity summed n-body matrix element can be written as

$$|\mathcal{M}_n|^2 = {}_n\langle 1, \ldots, n | 1, \ldots, n \rangle_n \tag{2.2.2}$$

where $|1, \ldots, n\rangle_n$ is a vector in colour + helicity space (see Appendix A for a more thorough explanation on notation).

2.2.1 Soft limits

Consider a tree-level matrix element $|\mathcal{M}_{n+1}|^2$ with a final-state gluon j. Note that reduced matrix elements associated with taking a quark soft have to vanish due to violation of quark number, therefore only gluons are considered. The limit of the soft gluon with momentum p_j can be parametrised as

$$p_j^\mu = \lambda q^\mu \,, \quad \lambda \to 0 \,, \tag{2.2.3}$$

where q^μ is an arbitrary four-vector and λ is a scale parameter. In this limit the matrix element can be written as

$$|\mathcal{M}_{n+1}|^2 \to -\frac{1}{\lambda^2} 8\pi\mu^{2\epsilon}\alpha_s \sum_i \frac{1}{p_i q} \sum_{k\neq i} \frac{p_k p_i}{(p_i+p_k)q}$$
$$ {}_n\langle 1,\ldots,j-1,j+1,\ldots,n+1|\boldsymbol{T}_k\cdot\boldsymbol{T}_i|1,\ldots,j-1,j+1,\ldots,n+1\rangle_n\,, \tag{2.2.4}$$

where terms less singular than $1/\lambda^2$ have been neglected. The reduced matrix element represented by the colour + helicity vector inner product is obtained by removing the soft gluon, j, from the $(n+1)$-body matrix element. The indices i, j, and k label partons involved in the factorisation process: i is the emitter parton, j is the emitted parton, and k is a parton accounting for the colour correlations. The scale μ can be identified as the renormalisation scale, and $\boldsymbol{T}_i$ is the colour-charge operator. These operators act on the colour space in the reduced matrix element to give rise to matrices, hence this factorisation is not exact. The properties of these operators are given in more detail in Appendix A.

The matrix elements in Eq. (2.2.4) are unambiguously defined only when momentum conservation is fulfilled. This is only true in the strict $\lambda = 0$ limit. Away from the limit, care has to be taken to conserve momentum conservation and keep the relevant partons on-shell. This can be done through the use of momentum mappings [51, 53].

2.2.2 Collinear limits

For the collinear limit, consider partons i and j in the matrix element $|\mathcal{M}_{n+1}|^2$. Their momenta can be decomposed as

$$\begin{aligned} p_i^\mu &= zp^\mu + k_\perp^\mu - \frac{k_\perp^2}{z}\frac{n^\mu}{2p\cdot n} \\ p_j^\mu &= (1-z)p^\mu - k_\perp^\mu - \frac{k_\perp^2}{1-z}\frac{n^\mu}{2p\cdot n}\,, \end{aligned} \tag{2.2.5}$$

where p^μ denotes the collinear direction of the two partons, $k_\perp^\mu$ specifies the transverse direction perpendicular to the collinear direction ($p\cdot k_\perp = 0$), and n^μ is an auxiliary vector satisfying the conditions $n^2 = 0$ and $k_\perp \cdot n = 0$. z is the fraction of momenta carried away from the collinear momentum by parton i.

The collinear limit can then be defined as

$$2p_i p_j = s_{ij} = -\frac{k_\perp^2}{z(1-z)}\,, \quad k_\perp \to 0\,. \tag{2.2.6}$$

In this limit the matrix element can be written as

$$|\mathcal{M}_{n+1}|^2 \to \frac{1}{p_i p_j} 4\pi\mu^{2\epsilon}\alpha_s \, {}_n\langle 1, \ldots, ij, \ldots, n+1|\hat{P}_{ij}(z, k_\perp)|1, \ldots, ij, \ldots, n+1\rangle_n\,, \tag{2.2.7}$$

where terms less singular than $1/k_\perp^2$ have been neglected. This reduced matrix element is obtained by replacing the partons i and j with a single parton ij, as explicitly shown in the reduced matrix element. This composite parton carries the momentum p^μ and suitable quantum numbers depending on the partons i and j. For example, if i = quark and j = gluon, then ij = quark, or if i = quark and j = antiquark, then ij = gluon. $\hat{P}_{ij}$ are the d-dimensional Altarelli-Parisi splitting functions that depend on the momentum fraction z for the splitting $ij \to i + j$ and the transverse momentum $k_\perp$. Each splitting function is a matrix acting on the spin indices of ij. Due to these spin correlations the reduced matrix element does not factorise from the splitting functions exactly. $\hat{P}_{ij}$ become the more recognisable Altarelli-Parisi splitting functions in Eq. (1.3.3) once spin-averaged and the $\epsilon \to 0$ limit is taken.

In general, it is possible for a final-state parton i to become collinear with an initial-state parton a. This is described by the splitting process $a \to ai + i$. In this case only momenta p_i needs to be modified as

$$p_i^\mu = (1-x)p_a^\mu + k_\perp^\mu - \frac{k_\perp^2}{1-x}\frac{n^\mu}{2p_a \cdot n}\,, \tag{2.2.8}$$

with the collinear limit defined as

$$2p_i p_a = s_{ia} = -\frac{k_\perp^2}{1-x}\,, \quad k_\perp \to 0\,. \tag{2.2.9}$$

The analogous expression of Eq. (2.2.7) for initial-state splitting is

$$|\mathcal{M}_{n+1}|^2 \to \frac{1}{x}\frac{1}{p_i p_a} 4\pi\mu^{2\epsilon}\alpha_s\, {}_n\langle 1,\ldots,n+1;ai,\ldots|\hat{P}_{ai}(x,k_\perp)|1,\ldots,n+1;ai,\ldots\rangle_n\,, \tag{2.2.10}$$

where the replacement of partons a and i have been made explicit by the presence of the parton ai in the colour + helicity vectors. The specific type of parton ai depends upon the types of a and i.

Similar to Eq. (2.2.4) where the factorisation was only true in the strict soft limit, Eqs. (2.2.7) and (2.2.10) are only true in the strict collinear limit. Away from these limits, care has to be taken to conserve momenta via the use of momenta mappings.

2.2.3 Factorisation of colour-ordered amplitudes

The factorisation formulae expressed in Eqs. (2.2.4), (2.2.7), and (2.2.10) were not exact because of colour and spin correlations. It can be shown that the factorisation of matrix elements in the soft and collinear limits becomes exact once the colour structure of the gauge group is separated from the kinematics.

In general, any QCD amplitude can be colour-decomposed, that is the colour structure is separated from the kinematics. Consider the process $e^+e^- \to q\bar{q} + n\ g$[6], the amplitude can be written as a product of hadronic and leptonic currents [54]

$$\mathcal{M}(q_1,\bar{q}_2;1,\ldots,n) = \hat{\mathcal{S}}_\mu^{n+2}(q_1;1,\ldots,n;\bar{q}_2)V^\mu\,, \tag{2.2.11}$$

where V^μ is the leptonic current and the hadronic current is

$$\hat{\mathcal{S}}_\mu^{n+2}(q_1;1,\ldots,n;\bar{q}_2) = \mathrm{i}eg_s^n \sum_{P(1,\ldots,n)} (T^{a_1}\ldots T^{a_n})_{c_1c_2} S_\mu(q_1;1,\ldots,n;\bar{q}_2)\,. \tag{2.2.12}$$

In this expression we have the electromagnetic gauge coupling e and the strong gauge coupling g_s appearing. The colour structure has been factorised into a product of fundamental group generators where the indices $a_i \in \{1,\ldots,N_c^2-1\}$ and $c_i \in$

[6] For the scope of this thesis it is sufficient to consider colour decomposition of electron-positron annihilation into a single quark pair plus gluons, and not the more general case of multiple quark flavours in the final-state.

$\{1,\ldots,N_c\}$. This leaves the colour-ordered partial amplitude $S_\mu(q_1;1,\ldots,n;\bar{q}_2)$ depending only on kinematic variables. In this partial amplitude, the gluons are emitted in an ordered fashion from the quarks, meaning the quarks have a fixed position in the partial amplitude. The sum over $P(1,\ldots,n)$ represents the sum over all permutations of gluon emissions which accounts for all Feynman diagrams and colour structures.

Upon squaring the amplitude in Eq. (2.2.11), we get

$$\left|\hat{\mathcal{S}}_\mu^{n+2}V^\mu\right|^2 = e^2\left(\frac{g_s^2 N_c}{2}\right)^n\left(\frac{N_c^2-1}{N_c}\right)\sum_{P(1,\ldots,n)}\left(\left|S_\mu(q_1;1,\ldots,n;\bar{q}_2)V^\mu\right|^2+\mathcal{O}\left(\frac{1}{N_c^2}\right)\right), \tag{2.2.13}$$

where the sub-leading colour terms proportional to $1/N_c^2$ have been omitted. The left-most term in the sum is the leading colour term and is the dominant term in the colour expansion.

With the amplitude written in terms of the colour-ordered partial amplitudes, it is now possible to factorise Eq. (2.2.13) exactly in the soft and collinear limits. Consider the limit where a final-state gluon j is soft, we have

$$\left|S_\mu(q_1;1,\ldots,i,j,k,\ldots,n;\bar{q}_2)V^\mu\right|^2 \to S_{ijk}\left|S_\mu(q_1;1,\ldots,i,k,\ldots,n;\bar{q}_2)V^\mu\right|^2, \tag{2.2.14}$$

where the factor

$$S_{ijk} = 4\frac{s_{ik}}{s_{ij}s_{jk}}, \tag{2.2.15}$$

is the well-known eikonal factor. We see that the colour-ordered amplitude on the RHS of Eq. (2.2.14) has gluon j removed but the ordering of all hard partons remain unchanged.

In the collinear limit where partons i and j become collinear to form parton k, the matrix element factorises as

$$\left|S_\mu(q_1;1,\ldots,i,j,\ldots,n;\bar{q}_2)V^\mu\right|^2 \to \frac{2}{s_{ij}}P_{ij}(z)\left|S_\mu(q_1;1,\ldots,k,\ldots,n;\bar{q}_2)V^\mu\right|^2, \tag{2.2.16}$$

where $P_{ij}(z)$ are the spin-averaged Altarelli-Parisi splitting functions with the colour factors removed. In this expression it is understood that the indices i, j, and k have to be self-consistent for the different types of partonic splittings. For partons which are not colour connected (are not neighbouring partons in the colour-ordered amplitude), there will be no singular behaviour as $s_{ij} \to 0$.

In Eqs. (2.2.14) and (2.2.16), the colour-ordered amplitude factorises exactly into a colour-ordered amplitude with one parton removed, multiplied by a universal singular factor. This singular factor depends on the unresolved parton and the two neighbouring hard particles. Interpreting this as the two hard particles forming an antenna which radiates the unresolved parton gives rise to the antenna functions which will be discussed further in Section 2.5.

2.2.4 One-loop matrix element factorisation

The discussion of matrix element factorisation so far has been focused on tree-level matrix elements. It has been shown that one-loop colour-ordered amplitudes also factorise in the soft and collinear limits [55–58]. At the one-loop level, there are new universal singular functions and the factorisation formulae are modified. The structure of these modified formulae are of the form: tree-level splitting function multiplied by a one-loop amplitude, plus a one-loop splitting function multiplied by a tree-level amplitude. The loop level of amplitude and splitting functions are denoted by superscripts, $^{(0)}$ for tree-level and $^{(1)}$ for one-loop level.

In the soft limit, a one-loop colour-ordered amplitude factorises as

$$M_{n+1}^{(1)}(\ldots,i,j,k,\ldots) \to S_{ijk}^{(0)}\, M_n^{(1)}(\ldots,i,k,\ldots) + S_{ijk}^{(1)}(\epsilon)\, M_n^{(0)}(\ldots,i,k,\ldots)\,, \quad (2.2.17)$$

where $S_{ijk}^{(0)}$ is the eikonal factor in Eq. (2.2.15), and $S_{ijk}^{(1)}(\epsilon)$ is the one-loop soft radiation function [59].

A similar factorisation formula for the collinear limit is

$$M_{n+1}^{(1)}(\ldots,i,j,\ldots) \to \frac{1}{s_{ij}}\left[P_{ij}^{(0)}(z)\, M_n^{(1)}(\ldots,k,\ldots) + P_{ij}^{(1)}(z,\epsilon)\, M_n^{(0)}(\ldots,k,\ldots)\right]\,, \quad (2.2.18)$$

where $P_{ij}^{(0)}(z)$ are the tree-level splitting functions in Eq. (1.3.3) and $P_{ij}^{(1)}(z,\epsilon)$ are the one-loop splitting functions [59].

In Section 2.5 we will give a brief overview of antenna functions where these one-loop factorisation formulae were used to obtain universal singular functions at the one-loop level for squared matrix elements. These antenna functions are then applied in the context of NLO QCD k-factor emulation for electron-positron annihilation in Chapter 6.

2.3 Subtraction

In Section 2.1.4 we discussed the structure of IR divergences and how the KLN theorem enforced the cancellation of IR divergences arising from the virtual corrections and real-emission matrix elements once integrated over soft and collinear regions of phase-space, at each order in perturbation theory.

Over the past few decades there has been a vast amount of research into devising methods to systematically isolate these singularities such that it is possible to make finite predictions of physical quantities. To tackle this problem, three main methods have been proposed: phase-space slicing [60–62], sector decomposition [63, 64], and subtraction [65].

By now the method of choice at NLO QCD is subtraction, and is the method we will focus on in this section. The main idea behind subtraction methods is to define local counterterms that exactly replicate the IR divergent behaviour of matrix elements. While there is not one single subtraction scheme that is universally used, the general form of a subtraction term must fulfil the requirements of replicating the matrix element behaviour in all IR singular limits, and be analytically integrable over the regions of phase-space corresponding to these IR limits.

To illustrate the idea behind the subtraction method, first consider the calculation of a LO partonic cross-section

$$\sigma^{\mathrm{LO}} = \int \mathrm{d}\Phi_n \, \mathcal{B}_n \,, \tag{2.3.1}$$

where $\mathcal{B}_n$ is the Born (tree-level) matrix element and Φ_n is the n-body phase-space. At the next order in perturbation theory, we have the NLO cross-section which receives contributions from the real and virtual corrections

$$\sigma^{\mathrm{NLO}} = \int \mathrm{d}\Phi_n \, \left[\mathcal{B}_n + \mathcal{V}_n\right] + \int \mathrm{d}\Phi_{n+1} \, \mathcal{R}_{n+1} \,, \tag{2.3.2}$$

where $\mathcal{V}_n$ is the virtual matrix element (renormalised to remove UV divergences as described in Section 2.1.1) which lives in the same n-body phase-space as the Born matrix element. The real-emission matrix element, $\mathcal{R}_{n+1}$, lives in the $(n+1)$-body phase-space, Φ_{n+1}, due to the emission of an additional external particle. The integrals over Φ_n and Φ_{n+1} are separately divergent but their sum is finite. To carry out a numerical calculation, it is therefore necessary to regulate these divergences to make them explicit. Using dimensional regularisation these divergences are mapped to poles in ϵ.

The motivation behind the subtraction method is that the divergences in Eq. (2.3.2) can be cancelled upon the insertion of a counterterm evaluated in Φ_{n+1}, $\mathcal{C}_{n+1}$, and an integrated counterterm evaluated in Φ_n, $\mathcal{I}_n$, such that the condition

$$\int \mathrm{d}\Phi_n \, \mathcal{I}_n - \int \mathrm{d}\Phi_{n+1} \, \mathcal{C}_{n+1} = 0 \,, \tag{2.3.3}$$

holds. Here phase-space factorisation is utilised: $\Phi_{n+1} \to \Phi_n \Phi_1$ when using an appropriate $3 \to 2$ momentum mapping [51, 66] such that

$$\mathcal{I}_n = \int \Phi_1 \, \mathcal{C}_{n+1} \,, \tag{2.3.4}$$

where Φ_1 is the one-parton phase-space leading to ϵ poles once $\mathcal{C}_{n+1}$ is integrated over. The counterterm $\mathcal{C}_{n+1}$ should be a proper pointwise approximation of $\mathcal{R}_{n+1}$ to cancel all IR divergences such that $\mathcal{R}_{n+1} - \mathcal{C}_{n+1}$ is integrable finite. Additionally, since the pole structure in $\mathcal{V}_n$ is identical to $\mathcal{R}_{n+1}$ but with an opposite sign, $\mathcal{V}_n + \mathcal{I}_n$

will be finite by construction. Inserting the subtraction terms into Eq. (2.3.2) we get

$$\sigma^{\mathrm{NLO}} = \int \mathrm{d}\Phi_n \left[\mathcal{B}_n + \mathcal{V}_n + \mathcal{I}_n\right] + \int \mathrm{d}\Phi_{n+1} \left[\mathcal{R}_{n+1} - \mathcal{C}_{n+1}\right] , \tag{2.3.5}$$

where each integral can now be carried out numerically in integer dimensions by taking the $\epsilon \to 0$ limit. This is possible as the integrals are all separately finite now.

The counterterm $\mathcal{C}_{n+1}$ has been kept general but specific examples of subtractions schemes include Catani-Seymour (CS) [51,67], Frixione-Kunszt-Signer (FKS) [68,69], and antenna subtraction [53, 54, 70].

Beyond NLO QCD, the algorithms available have not reached the maturity of the automated methods widely use at NLO. However, this is an active area of research. See Reference [71] for a review of methods that have been applied to NNLO QCD.

In the next sections we will describe in detail two sets of functions that are used to build subtraction terms: Catani-Seymour dipoles, and antenna functions. We are interested in the approximations of the matrix elements in the soft and collinear limits as these are universal and can be applied to any process, and not the subtraction terms themselves. These functions will become instrumental during our construction of matrix element emulators in Chapters 5, 6, and 7.

2.4 Catani-Seymour dipoles

Catani-Seymour dipoles introduced in Ref. [51] are process-independent functions that reproduce the IR singular behaviour of matrix elements. They depend on the momenta and quantum numbers of three partons in the real-emission phase-space. These three partons are identified by indices i, j, and k where i is the emitting parton, j is the unresolved emitted parton, and k is a spectator parton. In order to map out all the singular limits in a process, it is necessary to construct all permutations of the dipole functions since each dipole function only depends on three partons.

Dipole functions can be separated into four categories depending on whether the emitter and spectator are in the initial-state or the final-state, as illustrated in Figure 2.4. Namely, there are final-final (FF) dipoles, final-initial dipoles (FI), initial-final (IF) dipoles, and initial-initial (II) dipoles where the nomenclature refers to the emitter-spectator dipole. For electron-positron annihilations only FF dipoles are required since electrons and positrons do not carry any colour charge and so there is no initial-state radiation. However, for any hadronic collision the inclusion of the remaining dipoles are required to capture all IR-singular behaviour arising from the initial-state emissions.

In the following we will describe in detail the FF dipoles and the associated matrix element factorisation formula, but only give a brief description of the remaining dipoles as the structure of the terms and factorisation formulae generalise analogously.

2.4.1 Final-final dipoles

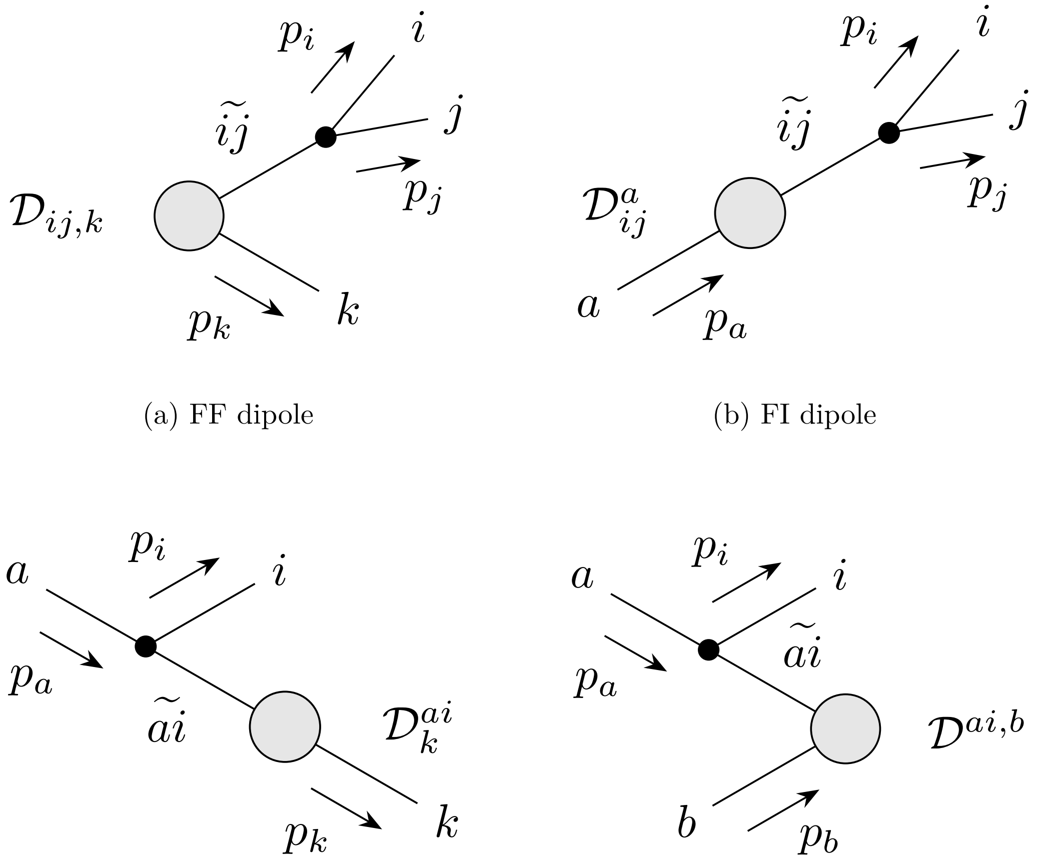
The utilisation of dipoles is encapsulated in the dipole factorisation formula where matrix elements in the limit $p_i p_j \to 0$ can be written as

$$_{n+1}\langle 1, \dots, n+1 | 1, \dots, n+1 \rangle_{n+1} = \sum_{i,j} \sum_{k \neq i,j} \mathcal{D}_{ij,k}(p_1, \dots, p_{n+1}) + \dots \,, \tag{2.4.1}$$

where terms not singular in the limit $p_i p_j \to 0$ are denoted by the ellipsis, and the dipole is

$$\mathcal{D}_{ij,k}(p_1, \dots, p_{n+1}) = -\frac{1}{2p_i p_j} \\ {}_{n}\langle 1, \dots, \widetilde{ij}, \dots, \tilde{k}, \dots, n+1 | \frac{\boldsymbol{T}_k \cdot \boldsymbol{T}_{ij}}{\boldsymbol{T}_{ij}^2} \boldsymbol{V}_{ij,k} | 1, \dots, \widetilde{ij}, \dots, \tilde{k}, \dots, n+1 \rangle_n \,, \tag{2.4.2}$$

where $\boldsymbol{T}_i$ are the colour-charge operators and $\boldsymbol{V}_{ij,k}$ is a matrix in the helicity space of the emitter embedding the IR divergent behaviour. The sum in Eq. (2.4.1) can be understood as summing over all possible three leg permutations to capture all the soft and collinear limits. The reduced matrix element on the RHS of Eq. (2.4.2) is obtained by replacing the partons i and j with a single parton $\widetilde{ij}$, and replacing



(a) FF dipole

(b) FI dipole

(c) IF dipole

(d) II dipole

Figure 2.4: Schematic diagrams of the four classes of Catani-Seymour dipoles, $\mathcal{D}$. The dipoles are named according to whether the emitter and spectator are in the initial (upper indices) or final-state (lower indices). Each dipole consists of a composite particle (denoted by tilde) that decays into two partons, and a spectator that recoils to conserve momentum. The grey blob represents the hard scattering process, with incoming and outgoing lines representing initial- and final-state partons, respectively. The black circle represents the splitting function within the dipole function which contains the divergent behaviour.

parton k with a parton $\tilde{k}$. The parton $\tilde{k}$ has all the same quantum numbers as k, whereas the partonic nature of $\widetilde{ij}$ depends on the specific splitting process (c.f. Section 2.2.2). The momenta of these particles are modified in the following way

$$\tilde{p}^\mu_{ij} = p^\mu_i + p^\mu_j - \frac{y_{ij,k}}{1-y_{ij,k}} p^\mu_k \,, \quad \tilde{p}^\mu_k = \frac{1}{1-y_{ij,k}} p^\mu_k \,, \tag{2.4.3}$$

where $y_{ij,k}$, the recoil parameter, is a dimensionless variable given as

$$y_{ij,k} = \frac{p_i p_j}{p_i p_j + p_i p_k + p_j p_k} \,. \tag{2.4.4}$$

Eqs. (2.4.3) and (2.4.4) are a 3 → 2 momenta mapping that maps $p_i + p_j + p_k \to \tilde{p}_{ij} + \tilde{p}_k$. The mapping ensures momentum conservation is maintained across all of phase-space and all particles are kept on-shell:

$$\begin{aligned} p^\mu_i + p^\mu_j + p^\mu_k &= \tilde{p}^\mu_{ij} + \tilde{p}^\mu_k \,, \\ \tilde{p}^2_{ij} &= \tilde{p}^2_k = 0 \,. \end{aligned} \tag{2.4.5}$$

The matrices $\boldsymbol{V}_{ij,k}$ are functions of $y_{ij,k}$ and the splitting variables $\tilde{z}$

$$\tilde{z}_i = \frac{p_i p_k}{p_i p_k + p_j p_k} \,, \quad \tilde{z}_j = \frac{p_j p_k}{p_i p_k + p_j p_k} = 1 - \tilde{z}_i \,. \tag{2.4.6}$$

These splitting variables are analogous to the z from Altarelli-Parisi splitting functions. $\boldsymbol{V}_{ij,k}$ acts on the spin indices of the composite particle $\widetilde{ij}$, but is independent of the type of the spectator. Making the spin-dependence on parton $\widetilde{ij}$ explicit (s and s' for $\widetilde{ij}$ =fermion, and μ and ν for $\widetilde{ij}$ =gluon) we list all the kernels here

$$\begin{aligned} \langle s|\boldsymbol{V}_{q_i g_j,k}|s'\rangle &= 8\pi\mu^{2\epsilon}\alpha_s C_F \left[\frac{2}{1-\tilde{z}_i(1-y_{ij,k})} - (1+\tilde{z}_i) - \epsilon\tilde{z}_j\right]\delta_{ss'} \,, \\ \langle \mu|\boldsymbol{V}_{q_i,\bar{q}_j,k}|\nu\rangle &= 8\pi\mu^{2\epsilon}\alpha_s T_R \left[-g^{\mu\nu} - \frac{2}{p_i p_j}(\tilde{z}_i p^\mu_i - \tilde{z}_j p^\mu_j)(\tilde{z}_i p^\nu_i - \tilde{z}_j p^\nu_j)\right] \,, \\ \langle \mu|\boldsymbol{V}_{g_i g_j,k}|\nu\rangle &= 16\pi\mu^{2\epsilon}\alpha_s C_A \Bigg[-g^{\mu\nu}\left(\frac{1}{1-\tilde{z}_i(1-y_{ij,k})} + \frac{1}{1-\tilde{z}_j(1-y_{ij,k})} - 2\right) \\ &\qquad + \frac{(1-\epsilon)}{p_i p_j}(\tilde{z}_i p^\mu_i - \tilde{z}_j p^\mu_j)(\tilde{z}_i p^\nu_i - \tilde{z}_j p^\nu_j)\Bigg] \,. \end{aligned} \tag{2.4.7}$$

A key feature of these kernels is that they smoothly interpolate between the soft and collinear limits, meaning they do not double count any limit. Additionally, they only contain divergences in the $p_i p_j \to 0$ limit and not for any other pair of momenta. In the soft and collinear limits, the dipole function $\mathcal{D}_{ij,k}$ correctly reproduces the matrix element factorisation behaviour seen in Eqs. (2.2.4) and (2.2.7), respectively. In particular, $\boldsymbol{V}_{ij,k}$ becomes proportional to the eikonal factor and Altarelli-Parisi splitting functions in these respective limits.

In this thesis we will be focussing on the emulation of colour- and spin-averaged matrix elements. Therefore the spin-indices are not explicitly available. It is possible to average over the spin indices in Eq. (2.4.7) to obtain the spin-averaged splitting functions

$$\begin{aligned}
\langle \boldsymbol{V}_{q_i g_j,k} \rangle &= 8\pi\mu^{2\epsilon}\alpha_s C_F \left[\frac{2}{1-\tilde{z}_i(1-y_{ij,k})} - (1+\tilde{z}_i) - \epsilon\tilde{z}_j\right] , \\
\langle \boldsymbol{V}_{q_i \bar{q}_j,k} \rangle &= 8\pi\mu^{2\epsilon}\alpha_s T_R \left[1 - \frac{2\tilde{z}_i\tilde{z}_j}{1-\epsilon}\right] , \\
\langle \boldsymbol{V}_{g_i g_j,k} \rangle &= 16\pi\mu^{2\epsilon}\alpha_s C_A \left[\frac{1}{1-\tilde{z}_i(1-y_{ij,k})} + \frac{1}{1-\tilde{z}_j(1-y_{ij,k})} - 2 + \tilde{z}_i\tilde{z}_j\right] . \qquad (2.4.8)
\end{aligned}$$

2.4.2 Final-initial dipoles

For the case of final-state emitter and initial-state spectator we have the dipole factorisation formula

$$\begin{aligned}
{}_{n+1}\langle 1,\ldots,n+1;a,\ldots|1,\ldots,n+1;a,\ldots\rangle_{n+1} = \\
\sum_{i,j}\sum_{k\neq i,j} \mathcal{D}_{ij,k}(p_1,\ldots,p_{n+1};p_a,\ldots) \qquad (2.4.9) \\
+\sum_{i,j}\sum_{a} \mathcal{D}_{ij}^a(p_1,\ldots,p_{n+1};p_a,\ldots) + \ldots ,
\end{aligned}$$

where the new dipole contribution appears on the second line. The initial-state parton is denoted by a. The dipole term $\mathcal{D}_{ij}^a$ is given by

$$\begin{aligned}
\mathcal{D}_{ij}^a(p_1,\ldots,p_{n+1};p_a,\ldots) = -\frac{1}{2p_ip_j}\frac{1}{x_{ij,a}} \\
{}_n\langle 1,\ldots,\widetilde{ij},\ldots,n+1;\tilde{a},\ldots|\frac{\boldsymbol{T}_a\cdot\boldsymbol{T}_{ij}}{\boldsymbol{T}_{ij}^2}\boldsymbol{V}_{ij}^a|1,\ldots,\widetilde{ij},\ldots,n+1;\tilde{a},\ldots\rangle_n .
\end{aligned}$$

(2.4.10)

where $\widetilde{ij}$ is the composite emitter in the final-state, and the spectator $\tilde{a}$ is in the initial-state. The momentum mapping for $p_i + p_j + p_a \to \tilde{p}_{ij} + \tilde{p}_a$ is given as

$$\tilde{p}_{ij}^\mu = p_i^\mu + p_j^\mu - (1 - x_{ij,a}) p_a^\mu \,, \quad \tilde{p}_a^\mu = x_{ij,a}\, p_a^\mu \,, \tag{2.4.11}$$

such that momentum conservation and on-shell conditions are met:

$$\begin{aligned} &p_i^\mu + p_j^\mu - p_a^\mu = \tilde{p}_{ij}^\mu - \tilde{p}_a^\mu \,, \\ &\tilde{p}_{ij}^2 = \tilde{p}_a^2 = 0 \,. \end{aligned} \tag{2.4.12}$$

The recoil parameter $x_{ij,a}$ and splitting variables $\tilde{z}$ are given by

$$\begin{aligned} x_{ij,a} &= \frac{p_i p_a + p_j p_a - p_i p_j}{(p_i p_a + p_j p_a)} \,, \\ \tilde{z}_i &= \frac{p_i p_a}{p_i p_a + p_j p_a} \,, \quad \tilde{z}_j = \frac{p_j p_a}{p_i p_a + p_j p_a} = 1 - \tilde{z}_i \,. \end{aligned} \tag{2.4.13}$$

The spin-averaged splitting functions are given as

$$\begin{aligned} \langle \boldsymbol{V}_{q_i g_j, a} \rangle &= 8\pi \mu^{2\epsilon} \alpha_s C_F \left[\frac{2}{1 - \tilde{z}_i + (1 - x_{ij,a})} - (1 + \tilde{z}_i) - \epsilon \tilde{z}_j \right] \,, \\ \langle \boldsymbol{V}_{q_i \bar{q}_j, a} \rangle &= 8\pi \mu^{2\epsilon} \alpha_s T_R \left[1 - \frac{2 \tilde{z}_i \tilde{z}_j}{1 - \epsilon} \right] \,, \\ \langle \boldsymbol{V}_{g_i g_j, a} \rangle &= 16\pi \mu^{2\epsilon} \alpha_s C_A \left[\frac{1}{\tilde{z}_j + (1 - x_{ij,a})} + \frac{1}{\tilde{z}_i + (1 - x_{ij,a})} - 2 + \tilde{z}_i \tilde{z}_j \right] \,. \end{aligned} \tag{2.4.14}$$

2.4.3 Initial-final dipoles

For the case of initial-state emitter and final-state spectator, the dipole factorisation formula is given as

$${}_{n+1}\langle 1, \ldots, n+1; a | 1, \ldots, n+1; a \rangle_{n+1} = \sum_{a,i} \sum_{k \neq i} \mathcal{D}_k^{ai}(p_1, \ldots, p_{n+1}; p_a) + \ldots \,, \tag{2.4.15}$$

where there is only one initial-state parton. The dipole is given by

$$\begin{aligned} \mathcal{D}_k^{ai}(p_1, \ldots, p_{n+1}; p_a) &= -\frac{1}{2 p_a p_i} \frac{1}{x_{ik,a}} \\ &\quad {}_n\langle 1, \ldots, \tilde{k}, \ldots, n+1; \widetilde{ai} | \frac{\boldsymbol{T}_k \cdot \boldsymbol{T}_{ai}}{\boldsymbol{T}_{ai}^2} \boldsymbol{V}_k^{ai} | 1, \ldots, \tilde{k}, \ldots, n+1; \widetilde{ai} \rangle_n \,. \end{aligned} \tag{2.4.16}$$

where the emitter is the initial-state parton $\widetilde{ai}$ and the spectator is the final-state parton $\tilde{k}$. The momenta mapping for $p_a + p_i + p_k \to \tilde{p}_{ai} + \tilde{p}_k$ is

$$\tilde{p}^\mu_{ai} = x_{ik,a}\, p^\mu_a \,, \quad \tilde{p}^\mu_k = p^\mu_k + p^\mu_i - (1 - x_{ik,a}) p^\mu_a \,, \tag{2.4.17}$$

such that

$$\begin{aligned} &p^\mu_i + p^\mu_k - p^\mu_a = \tilde{p}^\mu_k - \tilde{p}^\mu_{ai} \,, \\ &\tilde{p}^2_{ai} = \tilde{p}^2_k = 0 \,. \end{aligned} \tag{2.4.18}$$

Note that the momentum $\tilde{p}_{ai}$ is parallel to p_a, therefore the modified spectator momentum $\tilde{p}_k$ has to recoil the transverse momentum. This leads to $x_{ik,a}$ acting as a splitting variable. With $x_{ik,a}$ taking the role of $\tilde{z}$, there is another parameter, u_i, appearing in the splitting functions. These variables are given as

$$x_{ik,a} = \frac{p_k p_a + p_i p_a - p_i p_k}{p_i p_a + p_k p_a} \,, \quad u_i = \frac{p_i p_a}{p_i p_a + p_k p_a} \,. \tag{2.4.19}$$

The spin-averaged splitting functions are

$$\begin{aligned} \langle \boldsymbol{V}^{q_a g_i}_k \rangle &= \frac{n_s(q)}{n_s(\tilde{q})}\, 8\pi\mu^{2\epsilon}\alpha_s C_F \left[\frac{2}{1 - x_{ik,a} + u_i} + (1 + x_{ik,a}) - \epsilon(1 - x_{ik,a}) \right] , \\ \langle \boldsymbol{V}^{g_a \bar{q}_i}_k \rangle &= \frac{n_s(g)}{n_s(\tilde{q})}\, 8\pi\mu^{2\epsilon}\alpha_s T_R \left[1 - \frac{2x_{ik,a}(1 - x_{ik,a})}{1 - \epsilon} \right] , \\ \langle \boldsymbol{V}^{g_a g_i}_k \rangle &= \frac{n_s(g)}{n_s(\tilde{g})}\, 16\pi\mu^{2\epsilon}\alpha_s C_A \left[\frac{1}{1 - x_{ik,a} + u_i} + \frac{1 - x_{ik,a}}{x_{ik,a}} - 1 + x_{ik,a}(1 - x_{ik,a}) \right] , \\ \langle \boldsymbol{V}^{q_a q_i}_k \rangle &= \frac{n_s(q)}{n_s(\tilde{g})}\, 8\pi\mu^{2\epsilon}\alpha_s C_F \left[(1 - \epsilon) x_{ik,a} + 2\frac{1 - x_{ik,a}}{x_{ik,a}} \right] , \end{aligned} \tag{2.4.20}$$

where $n_s(a)$ is the number of polarisations of particle a. For fermions $n_s(q) = n_s(\bar{q}) = 2$ and for gluons $n_s(g) = d - 2$. In the limit $\epsilon \to 0$ all ratios of n_s cancel to give unity.

2.4.4 Initial-initial dipoles

In the case of two initial-state partons, a and b, there is an additional dipole contribution compared to the initial-final case.

The dipole factorisation is given by

$$\begin{aligned}
&{}_{n+1}\langle 1,\ldots,n+1;a,b|1,\ldots,n+1;a,b\rangle_{n+1} = \\
&\qquad\sum_{a,i}\sum_{k\neq i}\mathcal{D}_k^{ai}(p_1,\ldots,p_{n+1};p_a,p_b) \\
&\qquad+\sum_{a,i}\sum_{b\neq a}\mathcal{D}^{ai,b}(p_1,\ldots,p_{n+1};p_a,p_b)+\ldots\,,
\end{aligned} \tag{2.4.21}$$

where the new contribution is the dipole $\mathcal{D}^{ai,b}$ given by

$$\begin{aligned}
\mathcal{D}^{ai,b}(p_1,\ldots,p_{n+1};p_a,p_b) &= -\frac{1}{2p_ap_i}\frac{1}{x_{i,ab}} \\
&\quad {}_n\langle\tilde{1},\ldots,\widetilde{n+1};\widetilde{ai},b|\frac{\boldsymbol{T}_b\cdot\boldsymbol{T}_{ai}}{\boldsymbol{T}_{ai}^2}\boldsymbol{V}^{ai,b}|\tilde{1},\ldots,\widetilde{n+1};\widetilde{ai},b\rangle_n\,.
\end{aligned} \tag{2.4.22}$$

The initial-state parton $\widetilde{ai}$ is the emitter and the other initial-state parton b is the spectator. Notice that the momenta not involved in the dipole term have been modified with only parton b remaining unchanged. The momentum mapping for this case has $\tilde{p}_{ai}$ parallel with p_a and also modifies the momenta for all other final-state momenta (even non-QCD particles) k_j:

$$\begin{aligned}
\tilde{p}_{ai}^\mu &= x_{i,ab}\,p_a^\mu\,, \\
\tilde{k}_j^\mu &= k_j^\mu - \frac{2k_j\cdot(K+\widetilde{K})}{(K+\widetilde{K})^2}(K+\widetilde{K})^\mu + \frac{2k_j\cdot K}{K^2}\widetilde{K}^\mu\,, \\
x_{i,ab} &= \frac{p_ap_b-p_ip_a-p_ip_b}{p_ap_b}\,.
\end{aligned} \tag{2.4.23}$$

where K and $\widetilde{K}$ are the total momenta of the dipole before and after the mapping, respectively. They are given by

$$\begin{aligned}
K^\mu &= p_a^\mu + p_b^\mu - p_i^\mu\,, \\
\widetilde{K}^\mu &= \tilde{p}_{ai}^\mu + p_b^\mu\,.
\end{aligned} \tag{2.4.24}$$

This mapping conserves momentum and keeps mapped momenta on-shell

$$\begin{aligned}
&p_a^\mu + p_b^\mu - p_i^\mu - \sum_j k_j^\mu = \tilde{p}_{ai}^\mu + p_b^\mu - \sum_j \tilde{k}_j^\mu = 0\,, \\
&\tilde{p}_{ai}^2 = \tilde{k}_j^2 = 0\,.
\end{aligned} \tag{2.4.25}$$

The spin-averaged splitting functions are proportional to the Altarelli-Parisi splitting functions which we quote here for completeness

$$\begin{aligned}
\langle \boldsymbol{V}^{q_a q_i,b} \rangle &= \frac{n_s(q)}{n_s(\tilde{g})}\, 8\pi\mu^{2\epsilon}\alpha_s C_F \left[\frac{1+(1-x_{i,ab})^2}{x_{i,ab}}\right] , \\
\langle \boldsymbol{V}^{q_a g_i,b} \rangle &= \frac{n_s(q)}{n_s(\tilde{q})}\, 8\pi\mu^{2\epsilon}\alpha_s C_F \left[\frac{1+x_{i,ab}^2}{1-x_{i,ab}}\right] , \\
\langle \boldsymbol{V}^{g_a \bar{q}_i,b} \rangle &= \frac{n_s(g)}{n_s(\tilde{q})}\, 8\pi\mu^{2\epsilon}\alpha_s T_R \left[x_{i,ab}^2+(1-x_{i,ab})^2\right] , \\
\langle \boldsymbol{V}^{g_a g_i,b} \rangle &= \frac{n_s(g)}{n_s(\tilde{g})}\, 16\pi\mu^{2\epsilon}\alpha_s C_A \left[\frac{x_{i,ab}}{1-x_{i,ab}}+\frac{1-x_{i,ab}}{x_{i,ab}}+x_{i,ab}(1-x_{i,ab})\right] .
\end{aligned} \tag{2.4.26}$$

2.4.5 Master factorisation formula

The combination of all types of dipole contributions can be combined into a master dipole factorisation formula

$$\begin{aligned}
&{}_{n+1}\langle 1,\ldots,n+1|1,\ldots,n+1\rangle_{n+1} = \\
&\quad \sum_{i,j}\sum_{k\neq i,j}\mathcal{D}_{ij,k}(p_1,\ldots,p_{n+1};p_a,p_b)+\sum_{i,j}\sum_{a}\mathcal{D}_{ij}^{a}(p_1,\ldots,p_{n+1};p_a,p_b) \\
&\quad +\sum_{a,i}\sum_{k\neq i}\mathcal{D}_{k}^{ai}(p_1,\ldots,p_{n+1};p_a,p_b)+\sum_{a,i}\sum_{b\neq a}\mathcal{D}^{ai,b}(p_1,\ldots,p_{n+1};p_a,p_b)+\ldots ,
\end{aligned} \tag{2.4.27}$$

where it is understood that for a given partonic process, the matrix element in the different soft and collinear limits will be reproduced by summing over all the relevant dipoles, meaning that the dipole functions can be used as a set of functional behaviours for approximating the matrix elements for arbitrary SM processes. This will be put into practice in Chapters 5 and 7. The terms represented as ellipses in the dipole factorisation formulae are non-divergent in all of phase-space.

2.4.6 Treatment of massive partons

The dipoles described in this section so far have treated all partons to be massless. Since quarks are massive particles it is not always possible to treat them as massless. This is especially true for the top quark which has a mass of 172.69 GeV in the on-shell mass scheme [2], the most massive SM particle. In the case of partonic processes involving top quarks, it is therefore necessary to include mass effects into the dipole functions.

The universality of infrared divergent structure extends to massive quarks [72]. In Ref. [67] the authors derive dipole functions for massive partons which are used in Chapter 7, but will not be written here as the procedure of constructing the massive dipole functions is similar to their massless counterparts.

The extension to massive quarks necessitates the straightforward modification of the master dipole factorisation formula in Eq. (2.4.27) to replace any massless dipole with its massive counterpart whenever there is a massive parton in the dipole.

2.5 Antenna functions

Another set of functions that can be used to construct subtraction terms in NLO [53, 54] and NNLO QCD [52, 73] are the antenna functions. They are derived from physical matrix elements and so by construction contain the correct IR behaviour in the soft and collinear limits. More specifically, they are derived from colour-ordered matrix elements so they follow the colour-ordered factorisation properties outlined in Section 2.2.3. In the factorisation formulae, the colour-ordered matrix elements factorises exactly into a colour-ordered matrix element with one parton removed multiplied by a singular factor. Since this factorisation is exact, and the IR behaviour is universal, it is possible to extract these singular factors once and for all.

In contrast to the dipole functions, where there is an identified emitter, antenna functions can have the unresolved parton radiate from either of the two hard partons in the antenna. In this sense, an antenna is a linear combination of two dipoles.

The discussion in this section will be limited to the situation of an off-shell colour-neutral boson decaying to massless QCD partons, suitable for multi-jet production in electron-positron annihilation. Additionally, we assume that there will only ever be at most one unresolved parton in the final-state. The antenna formalism has been extended to more general cases than this [74–78].

Antenna functions are constructed from a ratio of colour-ordered matrix elements. For the case of one unresolved parton, the three-parton tree-level antenna function is given by

$$X_3^0(i,j,k) = S_{ijk,IK}\frac{|M_{n+1}^{(0)}(i,j,k)|^2}{|M_n^{(0)}(I,K)|^2} \tag{2.5.1}$$

where $|M_n^{(\ell)}|^2$ is the n-body colour-ordered matrix element at loop-level ℓ. $S_{ijk,IK}$ is a symmetry factor accounting for identical particles in the final-state and for the presence of multiple antennae in the two-parton process. I and K are hard partons forming a colour connected antenna that radiates particle j. The identities of i/I and k/K defines the specific class of the antenna function. There are three classes corresponding to the three underlying two-parton processes: quark-antiquark, quark-gluon and gluon-gluon. Therefore, there will be an antenna for every radiative correction to these two-parton processes. The physical matrix elements used to the derive the antenna functions for each class are as follows:

- Quark-antiquark: $\gamma^* \to q\bar{q}$ + (partons), the decay of a virtual photon into a quark-antiquark pair with QCD radiation from the quark pair [73].
- Quark-gluon: $\tilde{\chi} \to \tilde{g}g$ + (partons), the decay of a heavy neutralino into a gluino and gluon with QCD radiation from the gluon-gluino pair [79].
- Gluon-gluon: $H \to gg$ + (partons), the decay of a Higgs boson into a pair of gluons with QCD radiation from the gluon pair [80].

Along with the tree-level antennae, there are also the one-loop antennae. For the three-parton case these are given by

$$X_3^1(i,j,k) = S_{ijk,IK}\frac{|M_{n+1}^{(1)}(i,j,k)|^2}{|M_n^{(0)}(I,K)|^2} - X_3^0(i,j,k)\frac{|M_n^{(1)}(I,K)|^2}{|M_n^{(0)}(I,K)|^2}\,, \tag{2.5.2}$$

where X_3^1 are defined such that they are proportional to the one-loop singular functions mentioned in Section 2.2.4. Since the one-loop factorisation formulae, Eqs.

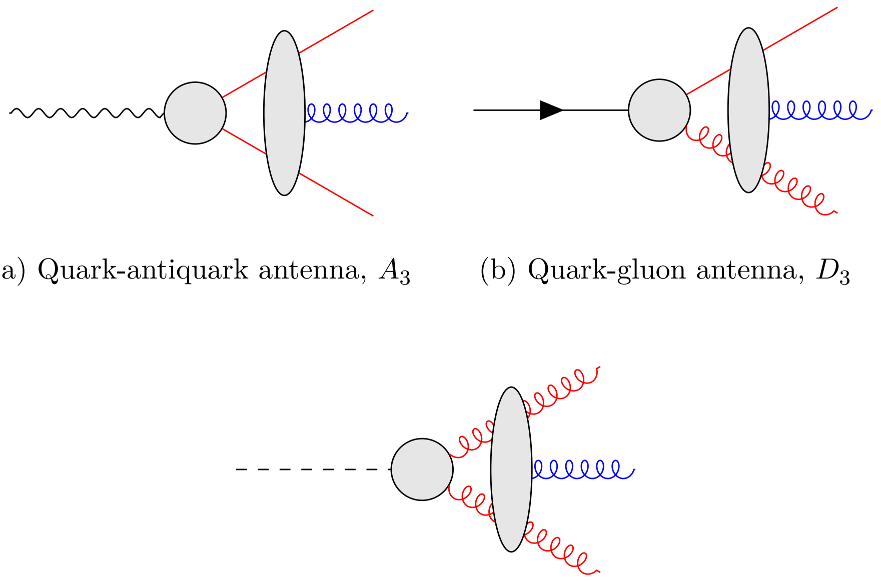
(2.2.17) and (2.2.18), are of the form $|M_{n+1}^{(1)}|^2 \to S^{(0)}|M_n^{(1)}|^2 + S^{(1)}|M_n^{(0)}|^2$, the term proportional to $S^{(0)}$, that is X_3^0, has to be removed in Eq. (2.5.2) to fulfil this requirement. The antenna functions are summarised in Table 2.1, where they are categorised by their class and radiative processes.

Class	Radiation	Antenna functions	
		Tree-level	One-loop
Quark-antiquark	$q\bar{q} \to qg\bar{q}$	A_3^0	A_3^1, $\tilde{A}_3^1$, $\hat{A}_3^1$
Quark-gluon	$qg \to qgg$	D_3^0	D_3^1, $\hat{D}_3^1$
	$qg \to qQ\bar{Q}$	E_3^0	E_3^1, $\tilde{E}_3^1$, $\hat{E}_3^1$
Gluon-gluon	$gg \to ggg$	F_3^0	F_3^1, $\hat{F}_3^1$
	$gg \to gq\bar{q}$	G_3^0	G_3^1, $\tilde{G}_3^1$, $\hat{G}_3^1$

Table 2.1: Three-parton antenna functions at tree-level, X_3^0, and one-loop level, X_3^1. The different antenna functions at one-loop level correspond to the leading colour (X_3^1), sub-leading colour ($\tilde{X}_3^1$), and closed quark loop ($\hat{X}_3^1$) contributions.

Another requirement in constructing a subtraction term is that the functions have to be analytically integrable over the unresolved phase-space. For one unresolved parton, this requirement is met with a suitable 3 → 2 momentum mapping that allows the $(n+1)$-body phase-space to factorise into a product of the mapped n-body phase-space and the antenna phase-space. The antenna phase-space is independent of the n-body phase-space and depends only on the momenta of partons i, j, k appearing in the antenna. In the case of antenna functions, the mapping of choice is the Kosower mapping [66] which maps $p_i + p_j + p_k \to p_I + p_K$. It reads

$$\begin{aligned} p_I &= x p_i + r p_j + y p_k \,, \\ p_K &= (1-x) p_i + (1-r) p_j + (1-y) p_k \,, \end{aligned} \tag{2.5.3}$$



(a) Quark-antiquark antenna, A_3

(b) Quark-gluon antenna, D_3

(c) Gluon-gluon antenna, F_3

Figure 2.5: Three-parton antenna functions where the loop order is implicitly illustrated by the grey circle, and the grey ellipse represents all diagrams that give rise to the given external states. The hard radiators are depicted in red, where as the unresolved parton is coloured in blue. External fermion lines have been drawn without an arrow to depict the equivalence of quark and antiquark antenna functions.

where the mapping variables are given as

$$
\begin{aligned}
s_{ijk} &= s_{ij} + s_{ik} + s_{jk}\,, \\
r &= \frac{s_{jk}}{s_{ij}s_{jk}}\,, \\
\rho &= \sqrt{1 + 4r(1-r)\frac{s_{ij}s_{jk}}{s_{ijk}s_{ik}}}\,, \\
x &= \frac{(1+\rho)s_{ijk} - 2rs_{jk}}{2(s_{ij}+s_{ik})}\,, \\
y &= \frac{(1-\rho)s_{ijk} - 2rs_{ij}}{2(s_{jk}+s_{ik})}\,.
\end{aligned}
\tag{2.5.4}
$$

This momentum mapping maintains momentum conservation and keeps the mapped partons on-shell

$$
\begin{aligned}
&p_i + p_j + p_k = p_I + p_K\,, \\
&p_I^2 = p_K^2 = 0\,.
\end{aligned}
\tag{2.5.5}
$$

In the following, we list the antenna functions relevant for multi-jet production in electron-positron annihilation with a single quark flavour, namely the partonic channel $e^-e^+ \to q\bar{q} + n\,g$. For this process we require the A, D, and F antenna functions, which are illustrated in Figure 2.5. The i, j, k indices will be fixed such that $q = 1$, $\bar{q} = 2$, and $g \in \{3, 4, 5\}$, in view of using the antenna functions with the momenta represented as an array with partons having fixed indices.

It will be useful to define the $\mathcal{P}oles$ and $\mathcal{F}inite$ operators to extract the singular and finite contributions from an antenna function. However, there are remaining finite terms from $\mathcal{P}oles(X)$ coming from the ϵ-expansion of the IR singularity operators given below. Therefore, care has to be taken to extract all the finite contributions from the antenna functions when trying to use them as part of an approximation for the finite part of the one-loop matrix element.

For the one-loop antenna functions, there are three different functions corresponding to the leading colour, sub-leading colour, and closed quark loop contributions. It is sufficient to consider the leading colour structure in this thesis as the remaining contributions are sub-leading. Ignoring sub-leading terms in colour is equivalent to taking the large N_c limit. For QCD where $N_c = 3$, neglecting sub-leading colour terms amounts to an approximately 10% effect.

The antenna functions have some functions in common, namely the singularity operators

$$\begin{aligned}
\mathbf{I}^{(1)}_{q\bar{q}}(\epsilon, s_{q\bar{q}}) &= -\frac{e^{\epsilon\gamma}}{2\Gamma(1-\epsilon)}\left[\frac{1}{\epsilon^2} + \frac{3}{2\epsilon}\right]\mathrm{Re}(-s_{q\bar{q}})^{-\epsilon}\,,\\
\mathbf{I}^{(1)}_{qg}(\epsilon, s_{qg}) &= -\frac{e^{\epsilon\gamma}}{2\Gamma(1-\epsilon)}\left[\frac{1}{\epsilon^2} + \frac{5}{3\epsilon}\right]\mathrm{Re}(-s_{qg})^{-\epsilon}\,,\\
\mathbf{I}^{(1)}_{gg}(\epsilon, s_{gg}) &= -\frac{e^{\epsilon\gamma}}{2\Gamma(1-\epsilon)}\left[\frac{1}{\epsilon^2} + \frac{11}{6\epsilon}\right]\mathrm{Re}(-s_{gg})^{-\epsilon}\,,\\
\mathbf{I}^{(1)}_{g\bar{q}}(\epsilon, s_{g\bar{q}}) &= \mathbf{I}^{(1)}_{qg}(\epsilon, s_{g\bar{q}})\,,
\end{aligned} \tag{2.5.6}$$

and dilogarithms which are embedded in the function

$$\begin{aligned}
R(y,z) =& \log(y)\log(z) - \log(y)\log(1-y) - \log(z)\log(1-z)\\
&+ \frac{\pi^2}{6} - \mathrm{Li}_2(y) - \mathrm{Li}_2(z)\,.
\end{aligned} \tag{2.5.7}$$

It will also be useful to introduce the variables $y_{ij} = s_{ij}/s_{ijk}$.

2.5.1 Quark-antiquark antenna functions

The quark-antiquark antenna functions are derived by normalising the colour-ordered QCD radiative corrections to $\gamma^* \to q\bar{q}$ at NNLO.

Three-parton tree-level antenna function

$$A_3^0(1_q,3_g,2_{\bar{q}}) = \frac{1}{s_{123}}\left(\frac{s_{13}}{s_{23}} + \frac{s_{23}}{s_{13}} + 2\frac{s_{12}s_{123}}{s_{13}s_{23}}\right) + \mathcal{O}(\epsilon) \tag{2.5.8}$$

Three-parton one-loop antenna function

$$\begin{aligned}\mathcal{P}oles\left(A_3^1(1_q,3_g,2_{\bar{q}})\right) &= 2\left(\mathbf{I}^{(1)}_{qg}(\epsilon,s_{13}) + \mathbf{I}^{(1)}_{qg}(\epsilon,s_{23})\right.\\ &\quad \left.-\mathbf{I}^{(1)}_{q\bar{q}}(\epsilon,s_{123})\right) A_3^0(1_q,3_g,2_{\bar{q}})\,, \qquad (2.5.9)\\ \mathcal{F}inite\left(A_3^1(1_q,3_g,2_{\bar{q}})\right) &= -\left(R(y_{13},y_{23}) + \frac{5}{3}\log y_{13} + \frac{5}{3}\log y_{23}\right) A_3^0(1_q,3_g,2_{\bar{q}})\\ &\quad + \frac{1}{s_{123}} + \frac{s_{12}+s_{23}}{2s_{123}s_{13}} + \frac{s_{12}+s_{13}}{2s_{123}s_{23}}\\ &\quad - \frac{s_{13}}{2s_{123}(s_{12}+s_{13})} - \frac{s_{23}}{2s_{123}(s_{12}+s_{23})}\\ &\quad + \frac{\log y_{13}}{s_{123}}\left(2 - \frac{1}{2}\frac{s_{13}s_{23}}{(s_{12}+s_{23})^2} + 2\frac{s_{13}-s_{23}}{s_{12}+s_{23}}\right)\\ &\quad + \frac{\log y_{23}}{s_{123}}\left(2 - \frac{1}{2}\frac{s_{13}s_{23}}{(s_{12}+s_{13})^2} + 2\frac{s_{23}-s_{13}}{s_{12}+s_{13}}\right)\,. \qquad (2.5.10)\end{aligned}$$

2.5.2 Quark-gluon antenna functions

The quark-gluon antenna functions are obtained by considering the QCD real radiation corrections in the process $\tilde{\chi} \to \tilde{g}g$.

Three-parton tree-level antenna function

$$\begin{aligned}D_3^0(1_q,3_g,4_g) = \frac{1}{s_{134}^2}&\left(\frac{2s_{134}^2 s_{14}}{s_{13}s_{34}} + \frac{2s_{134}^2 s_{13}}{s_{14}s_{34}} + \frac{s_{14}s_{34}+s_{34}^2}{s_{13}}\right.\\ &\left.+\frac{s_{13}s_{34}+s_{34}^2}{s_{14}} + \frac{2s_{13}s_{14}}{s_{34}} + 5s_{134} + s_{34}\right) + \mathcal{O}(\epsilon)\,.\end{aligned} \tag{2.5.11}$$

Three-parton one-loop antenna functions

$$\mathcal{P}oles\left(D_3^1(1_q,3_g,4_g)\right) = 2\left(\mathbf{I}_{qg}^{(1)}(\epsilon,s_{13}) + \mathbf{I}_{qg}^{(1)}(\epsilon,s_{14}) + \mathbf{I}_{gg}^{(1)}(\epsilon,s_{34}) \right.$$
$$\left. -2\mathbf{I}_{qg}^{(1)}(\epsilon,s_{134})\right) D_3^0(1_q,3_g,4_g)\,, \qquad (2.5.12)$$

$$\mathcal{F}inite\left(D_3^1(1_q,3_g,4_g)\right) = -\left(R(y_{13},y_{34}) + R(y_{14},y_{34}) + R(y_{13},y_{14}) + \frac{5}{3}\log y_{13}\right.$$
$$\left. + \frac{5}{3}\log y_{14} + \frac{11}{6}\log y_{34}\right) D_3^0(1_q,3_g,4_g) + \frac{1}{3s_{34}}\,.$$
$$(2.5.13)$$

2.5.3 Gluon-gluon antenna functions

The gluon-gluon antenna functions are obtained from the QCD real radiation corrections in the process $H \to gg$.

Three-parton tree-level antenna function

$$F_3^0(3_g,4_g,5_g) = \frac{2}{s_{345}^2}\left(\frac{s_{345}^2 s_{34}}{s_{35}s_{45}} + \frac{s_{345}^2 s_{35}}{s_{34}s_{45}} + \frac{s_{345}^2 s_{45}}{s_{34}s_{35}}\right.$$
$$\left. + \frac{s_{34}s_{35}}{s_{45}} + \frac{s_{34}s_{45}}{s_{35}} + \frac{s_{35}s_{45}}{s_{34}} + 4s_{345} + \mathcal{O}(\epsilon)\right)\,. \qquad (2.5.14)$$

Three-parton one-loop antenna functions

$$\mathcal{P}oles\left(F_3^1(3_g,4_g,5_g)\right) = 2\left(\mathbf{I}_{gg}^{(1)}(\epsilon,s_{34}) + \mathbf{I}_{gg}^{(1)}(\epsilon,s_{35}) + \mathbf{I}_{gg}^{(1)}(\epsilon,s_{45})\right.$$
$$\left. -2\mathbf{I}_{gg}^{(1)}(\epsilon,s_{345})\right) F_3^0(3_g,4_g,5_g)\,, \qquad (2.5.15)$$

$$\mathcal{F}inite\left(F_3^1(3_g,4_g,5_g)\right) = -\left(R(y_{34},y_{35}) + R(y_{35},y_{45}) + R(y_{34},y_{45})\right.$$
$$\left. + \frac{11}{6}\log y_{34} + \frac{11}{6}\log y_{35} + \frac{11}{6}\log y_{45}\right) F_3^0(3_g,4_g,5_g)$$
$$+ \frac{1}{3s_{34}} + \frac{1}{3s_{35}} + \frac{1}{3s_{45}} + \frac{1}{3s_{345}}\,. \qquad (2.5.16)$$

2.5.4 Limiting behaviour of antenna functions

Examining Eq. (2.5.8) more closely, it becomes clear that the correct IR behaviour is exhibited by the antenna function. The first two terms correspond to the gluon going collinear with the quark and antiquark, respectively. The final term encapsulates the singularity in the limit of the gluon going soft. The other antenna functions have similar limiting behaviour, collapsing to the single collinear and soft splitting functions in the relevant regions of phase-space.

Utilising this property of the antenna functions, and the fact that they naturally interpolate between the soft and collinear limits due to being derived from physical matrix elements, they can be combined in an ansatz for approximating matrix elements. The emulation of the NLO QCD k-factors for electron-positron annihilation is explored in Chapter 6.

Chapter 3

Monte Carlo Event Generators

At particle collider experiments, the collisions between incoming particles can lead to highly complex final-states with a large number of particles. In any collider involving QCD processes, the particles we observe are not the constituents of the proton, the quarks and gluons. Instead we observe hadrons which are the bound states of these partons due to the property of confinement at low energy scales.

In order to describe the entire process from the initial energetic collision to the production of the lower energy hadrons, it is customary to split up the process into stages characterised by their kinematic hardness (c.f. Section 1.3). The standard tools used to simulate these stages of an event are general-purpose Monte Carlo event generators, see Ref. [81] for a review. These are indispensable tools that help bridge the gap between theoretical predictions and experimental measurements with nearly all analyses made at the ATLAS and CMS experiments involving some form of event generator usage. General-purpose event generators such as Herwig [82, 83], Pythia [84, 85], and Sherpa [86, 87] all broadly share the same features at a high-level, but each framework carries subtle differences. Hence, it is common to use many different event generators in one analysis to obtain an estimate of the uncertainty due to the different approaches in each framework.

In the following we will motivate the usage of Monte Carlo methods and give a brief overview of the main stages involved in simulating an event. We will give particular attention to the concept of event unweighting which is required to compare probabilistically generated events to experimental results.

3.1 Monte Carlo integration

As seen in Section 1.4, the task of computing partonic cross-sections boils down to integrating the matrix elements over the final-state phase-space. Whilst it is possible to carry out these integrals analytically for the simplest cases, the integral quickly becomes intractable when the final-state multiplicity becomes large. For an n-body final-state the dimensionality of the integral scales as $d = 3n - 4$, where momentum conservation and on-shell conditions have already been applied to reduce the dimensionality. Therefore, instead of carrying out the integrals analytically, numerical methods become more attractive. The current prevailing method is Monte Carlo integration as the integration error has no scaling with the dimensionality of the integral, and so can be used for arbitrarily high multiplicity final-states with no performance penalty compared to lower multiplicities. The purpose of this chapter is to give a brief overview of Monte Carlo integration in the context of event generation, a more in-depth treatment of the specifics of Monte Carlo methods can be found in Ref. [88].

The basic idea of Monte Carlo integration is to approximate the integral of a function $f(\mathbf{x})$ by sampling it in d-dimensional parameter space points $\mathbf{x}$

$$I = \int_V \mathrm{d}\mathbf{x}\, f(\mathbf{x}) \approx I_N = V \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) = V \langle f \rangle_{\mathbf{x}}\,, \tag{3.1.1}$$

where the points $\mathbf{x}$ are randomly, uniformly sampled in the volume V. Given a sample of N points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in V$, $\langle f \rangle_{\mathbf{x}}$ denotes the average of f over these uniformly sampled points. The law of large numbers ensures that the estimator I_N will converge to the true value I for $N \to \infty$. As a result of the central limit theorem, the function evaluations $f(\mathbf{x}_i)$ will follow a Gaussian distribution with mean I_N, meaning an error estimate on the estimator is given by the standard error of the mean

$$\sigma_I = V\sqrt{\frac{\langle f^2\rangle_{\mathbf{x}} - \langle f\rangle_{\mathbf{x}}^2}{N-1}} \tag{3.1.2}$$

which falls as $1/\sqrt{N}$, with no dependence on d.

An algorithm that samples points $\mathbf{x}$ is called an integrator. To make use of the random numbers sampled, they need to be mapped to quantities that are useful in a simulation setting. The most commonly used mapping is to map the random numbers to four-momenta. An example of a simple integrator is the RAMBO algorithm [89, 90] which samples four-momenta space uniformly in a phase-space integration. Since the integrator now effectively samples in momentum space, we refer to the points generated as phase-space points and so the terms integrator and phase-space generator are used interchangeably.

Whilst Monte Carlo integration is independent of the dimensionality of the integral, meaning convergence is guaranteed for large enough N, there are variance reduction methods to more rapidly reduce the error on the estimator. These methods aim to reduce the numerator in Eq. (3.1.2).

One such method is importance sampling which aims to reduce the variance via a remapping of the uniformly sampled variables $\mathbf{x}$ to a more suitable non-uniform distribution. In the context of particle physics, this corresponds to using a mapping of random numbers to four-momenta that captures the distribution of the matrix elements and their divergences, i.e. samples more frequently in regions where the matrix element is large in magnitude. Integrators inspired by the pole structure of multi-parton QCD processes have been studied in Refs. [91, 92].

Another common variance reduction technique is stratified sampling which divides the integration volume into sub-volumes, or bins. The overall integral and variance are the sums of the partial results in each bin. The overall variance is minimised when the contributions from each bin are equal. This is achieved by sampling more from bins where the integral is rapidly fluctuating (and not necessarily large), and less from bins where the integral has fewer fluctuations.

In practice, state of the art event generators combine the concepts of importance sampling and stratified sampling with adaptive sampling [93, 94] and multichannelling methods [95, 96] to achieve faster rates of convergence.

3.2 Event unweighting

Once a suitable configuration of momenta has been found by the integrator, this phase-space point forms the basis of an event. The matrix element of interest can be evaluated at this phase-space point at the desired order in perturbation theory (see Section 3.3). Additionally, there will be an associated Jacobian from the mapping from random variables to the four-momenta. This Jacobian, along with the PDF weights and flux factors, are referred to as the phase-space weight, J_{PS}. Taking this phase-space weight together with the matrix element forms the event weight[7]

$$w = |\mathcal{M}|^2 J_{\text{PS}} \,. \tag{3.2.1}$$

The event weight can be interpreted as the probability of the event occurring at an experiment [97]. By generating a number of events in this manner a sample of weighted events is created. Cross-sections and distributions can then be calculated by histogramming these weights.

The generation of weighted events is generally undesirable as many events will have small weights with only a few events making sizeable contributions to predictions. Furthermore, these small weights still have to be run through expensive detector simulations, making the usage of weighted events highly inefficient. Therefore, it has become customary to generate events with equal relative weight, so-called unweighted events.

An additional reason to generate unweighted events is that experimentalists would like the probability of events occurring at colliders to be described only by their

[7] For the more sophisticated sampling methods employed in event generators there will be additional weights involved.

frequency and not with any additional weights. Since unweighted events have equal relative weight, they satisfy this criteria.

The process of generating unweighted events is known as unweighting and is typically carried out using a rejection sampling algorithm. Events are accepted or rejected by comparing the ratio w/w_{max} to a uniform random number, r, sampled in the interval $[0, 1]$, where w_{max} is the maximal event weight in the integration volume. Events are then accepted if $w/w_{\text{max}} > r$. This can be interpreted as converting the event weights into a probability of being accepted or rejected as small event weights will have a low probability of being accepted, whereas the opposite is true for large event weights. The rejection algorithm is outlined explicitly in Algorithm 1.

Algorithm 1: Rejection sampling for unweighting events

```
while unweighting do
    generate random phase-space point $\mathbf{x}$;
    evaluate event weight $w \leftarrow w(\mathbf{x})$;
    generate uniform random number $r \leftarrow \text{Random}(0,\ 1)$;
    if $w/w_{\text{max}} > r$ then
        return $\mathbf{x}$ and $\tilde{w} \leftarrow 1$;
    end
end
```

From this description, it is clear that the number of accepted events will generally be lower than the number of total events generated. This is encapsulated in the unweighting efficiency

$$\epsilon = \frac{N_{\text{accepted}}}{N_{\text{total}}}\,, \tag{3.2.2}$$

whose inverse is the average number of trial events required to obtain a single unweighted event. Unweighting efficiencies for high-multiplicity processes are often well below 1% [98, 99]. The development of methods to improve the efficiency are an active area of research [100, 101], wherein the authors explore an algorithm that adaptively splits up the integration region in order to systematically reduce the ratio of the maximum event weight to the average event weight. This leads to increased unweighting efficiency when carrying out the rejection sampling algorithm. These low efficiencies lead to most of the yearly event generator CPU budgets of LHC experiments to be spent on unweighting events [102].

An alternative to improving the unweighting efficiency is to accelerate the process of unweighting itself. This was explored in Ref. [103] wherein a novel two-stage unweighting procedure employing a surrogate[8] model of the exact event weight was used to decrease the amount of time spent unweighting events. This idea will be elaborated on in Chapter 7 where we build upon Ref. [103] by increasing the fidelity of the surrogate model.

3.3 Anatomy of an event

The simulation of an event can be split up into three main stages: the hard scattering process, followed by the parton shower, and finally hadronisation. This is motivated by the factorisation theorem (c.f. Section 1.3) which separates the high energy regimes from the low energy regimes.

Hard scattering

The first stage of an event simulation begins with the computation of the matrix element. The partonic channel of interest sets the multiplicity of the process, therefore informing the integrator on how to sample the phase-space accordingly.

As seen in Eq. (1.4.9), the exact matrix element is approximated by a perturbative series in a coupling parameter. By now it is standard to have at least the LO and NLO contributions in this expansion. The calculations of tree-level [104–109] and one-loop [110–123] matrix elements are now largely fully automated in NLO QCD and electroweak corrections. Many of the modern matrix element providers are supplied with a common generic interface to event generators, the Binoth Les Houches Accord [124, 125]. For more information on the machinery developed to

[8] The authors of Ref. [103] refer to their model approximating the event weight as a surrogate model, whereas in this thesis we have used the term emulator. We will use these two terms interchangeably henceforth.

compute matrix elements at tree-level and one-loop level required for SM predictions, see Ref. [126]. NNLO and N^{3}LO QCD corrections have been computed for select processes but have yet to reach the level of automation of NLO. For a review of recent advancements see Refs. [127, 128].

Matrix element calculations account for a large portion of the total event generation time [129]. Reducing the time spent in this stage of the event generation therefore represents a sizeable increase in the number of events generated given a fixed computing budget [102]. This thesis presents a novel technique to accelerate matrix element calculations by building accurate emulation models from modern machine learning techniques.

Parton shower

Fixed-order calculations of matrix elements are limited to relatively few final-state particles due to the complexity of analytical expressions and their phase-space integrals. They are also only accurate at high energy scales. For low transverse momentum emissions, large logarithms appearing at every order breaks the α_s perturbative expansion. To describe these kinematic regions, and to evolve from the fixed-order calculation to the non-perturbative regime where hadrons are formed, parton showers are employed in event generators.

Parton showers are implemented as algorithms that iteratively emit radiation (gluons and quarks in QCD parton showers) described by splitting functions. By repeating this procedure of emissions, parton showers produce a cascade of partons that naturally evolves the energy scale from the hard process to an infrared scale where non-perturbative effects begin to set in. In doing so, parton showers account for potentially large logarithms that arise in these kinematic regimes at all orders in the coupling parameter, this is known as resummation [81].

In event generators, fixed-order matrix elements and parton showers have to be utilised together to describe the many different observables at experiments. In order to combine the two stages of event generation in a consistent manner there are matching [130–132] and merging [133, 134] schemes.

Hadronisation

The simulation of the formation of hadrons in the low-energy regime of an event generator follows phenomenological models, which are non-perturbative and consist of many parameters fitted to data. These hadronisation models require the combination of quarks and antiquarks to form colourless hadrons. The two main classes of models used to simulate hadronisation are the cluster models [135, 136] and string models [137]. For more discussion on hadronisation see Ref. [138].

3.4 Estimation of theoretical uncertainties

The partonic cross-section, $\hat{\sigma}(\mu_F, \mu_R)$, has a dependence on both the factorisation scale and the renormalisation scale. Although these scales are unphysical, and so any observable should not depend on them, there is a residual dependence on them in fixed-order calculations due to missing higher order terms.

The most widely adopted approach to estimate the theoretical uncertainties associated with these missing higher order terms is to carry out a seven point scale variation. Typically in a calculation, $\mu_0 = \mu_F = \mu_R$ where μ_0 depends on the specifics of the process of interest. Taking the value of the partonic cross-section at this scale as the central value, the seven variations correspond to multiplying (μ_F, μ_R) by the factors

$$S \in \{(1,1), (1/2, 1/2), (2,2), (1/2, 1), (1, 1/2), (2,1), (1,2)\} \ , \tag{3.4.1}$$

and re-evaluating the cross-section at these scales. The deviations from the central cross-section value forms an estimate of the uncertainty in the perturbatively computed cross-section, when compared to the central cross-section value. This can be written as

$$\hat{\sigma}(\mu_F, \mu_R) = \hat{\sigma}(\mu_0, \mu_0) \pm \delta\,, \tag{3.4.2}$$

where δ is determined from the scale variations. However, there is no single, well agreed upon definition of δ [139]. The rationale behind taking this approach is that δ is approximately the same order of magnitude as the true missing higher order terms. This is true if the size of the perturbative coefficients remains similar, which would be violated if, for example, a new important channel opens up by increasing the perturbative order.

Chapter 4

Machine learning in high energy physics

4.1 Rise of machine learning in high energy physics

In high energy physics, where analyses depend on making use of large computing clusters and international grid efforts [140], advancements in technology and more efficient methods are required to keep up with the needs of the experimental and theoretical communities. However, the computational resources needed are at risk of outpacing the growth in these research areas [129], especially with HL-LHC projected to begin operating at the end of this decade [141]. Therefore new techniques and algorithms from machine learning have been adopted to augment the ongoing efforts in high energy physics. Machine learning is a category of artificial intelligence concerned with the design of algorithms that automates learning from data. By now, machine learning is ubiquitous in particle physics analysis, from collection of data from experiments, all the way to novel applications on the theory side.

Around the turn of the last decade the culmination of advancement in hardware and training algorithms led to an explosion in research in the machine learning community, specifically in the utilisation of very large neural networks. The accuracy of these deep neural networks greatly outperformed the previous state of the art [142,143] and lead to the widespread use of neural networks across a range of tasks [144]. This led to the rise of programming frameworks such as scikit-learn [145], TensorFlow [146], PyTorch [147], and XGBoost [148] to name a few which implement neural networks, alongside other mainstay algorithms such as decisions trees. These algorithms are now extremely commonplace in many domains, including in particle physics.

Collisions at the LHC can produce hundreds of particles with complex final-state configurations. This leads to the design of the ATLAS and CMS experiments to contain of the order 100 million detection elements in an attempt to disentangle these events. With these large arrays, petabytes of data are recorded every year across all LHC experiments. This presents a practical challenge for data collection where machine learning could provide a solution. Indeed, boosted decision trees have been used to enhance triggers [149] to accept or reject data before being saved to disk.

In view of the upgrades at the LHC and the upcoming HL-LHC providing unprecedented integrated luminosity, it is important that the generation of simulated samples is accelerated. This prompted the explosion of research in machine learning applications on the theory side. Studies have been carried out on many stages of the event generation process. To name just a few examples of active research areas: phase-space sampling [150–154], matrix element modelling [4,5,155–157], hadronisation modelling [158–160], and end-to-end event generation [99,161–164]. For a more complete overview of active research, a living review that is archiving advancements in the field is available at Ref. [165]. For more traditional reviews, see Refs. [166–168].

The research carried out in these areas have shown promising results but a challenge that remains is to interface these novel machine learning methods to existing event generators for real-world use. There is scepticism regarding the use of machine learning algorithms in production due to their lack of interpretability [169], as well as their use as black-boxes which precludes predictability [170]. Along similar lines, the quantification of the uncertainties associated with these algorithms are not well defined [171].

In this chapter we give an overview of neural networks: how they are constructed, how they are trained, and how their parameters are optimised. We will then proceed to discuss their application to modelling matrix elements with a review of the current state of the art methods.

4.2 Neural networks

Neural networks are a machine learning algorithm inspired vaguely by the structure of the human brain. That is to say they are an interconnected network of neurons used for analysing data. Here, we will introduce the densely-connected neural network which will be the main workhorse algorithm used in this thesis, and frame the discussion around a regression task. Namely, how to use sampled training data from a target function $g(\mathbf{x})$ to create a surrogate model with a neural network algorithm.

In a machine learning context, the fitting of a model to data is termed training the model. Once the model is fitted, it is customary to use an unbiased dataset that is distinct from the training set, the test set, to evaluate the final model performance.

4.2.1 Model of a neuron

In this context, a neuron is described by

$$y = \phi(\mathbf{w}^T\mathbf{x} + b) = \phi_\theta(\mathbf{x}) \tag{4.2.1}$$

where the model inputs, $\mathbf{x}$, are multiplied by the model weights, $\mathbf{w}$, before being summed with a bias term, b. Collectively, the model weights and biases are the model parameters, θ. This combination of terms is then modified by an activation function, ϕ, which is carefully chosen to perform a non-linear transformation. The

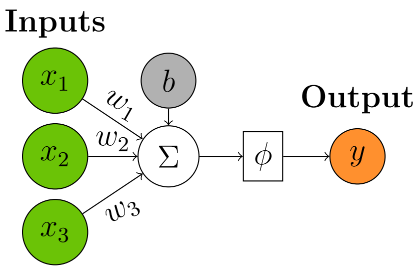


Figure 4.1: A schematic diagram of a neuron. The model inputs $\mathbf{x}$ and weights $\mathbf{w}$ are vectors, whereas the bias b is a scalar. The weighted inputs are summed with the bias before passing through an activation function to give the scalar output y.

Activation	Functional form	Gradient
Linear	$f(x) = x$	$f'(x) = 1$
Sigmoid	$f(x) = \dfrac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh	$f(x) = \tanh(x)$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0\,, & \text{for} \quad x \leq 0 \\ x\,, & \text{for} \quad x > 0 \end{cases}$	$f'(x) = \begin{cases} 0\,, & \text{for} \quad x \leq 0 \\ 1\,, & \text{for} \quad x > 0 \end{cases}$ [9]
swish	$f(x) = \dfrac{x}{1+e^{-x}}$	$f'(x) = \dfrac{1}{x} f(x) \left[(1+x) - f(x)\right]$

Table 4.1: Functional forms of common activation functions and their gradients.

notation ϕ_θ represents an activation function acting on $\mathbf{x}$ which has been combined with parameters θ. A neuron is illustrated in Figure 4.1.

Some of the more well-known activation functions are the sigmoid function, hyperbolic tangent, and the rectified linear unit (ReLU) [172]. A less familiar class of activation functions are the sigmoid linear units (SiLU) [173], of which swish [174] is an example. The functional forms of these functions and their gradients are summarised in Table 4.1. There is no single activation function that is the best for any given task, and it is common practice to choose activation functions on a case-by-case basis.

[9] The gradient of ReLU is not defined at $x = 0$ but for a numerical implementation, defining it to be 0 at this point is sufficient.

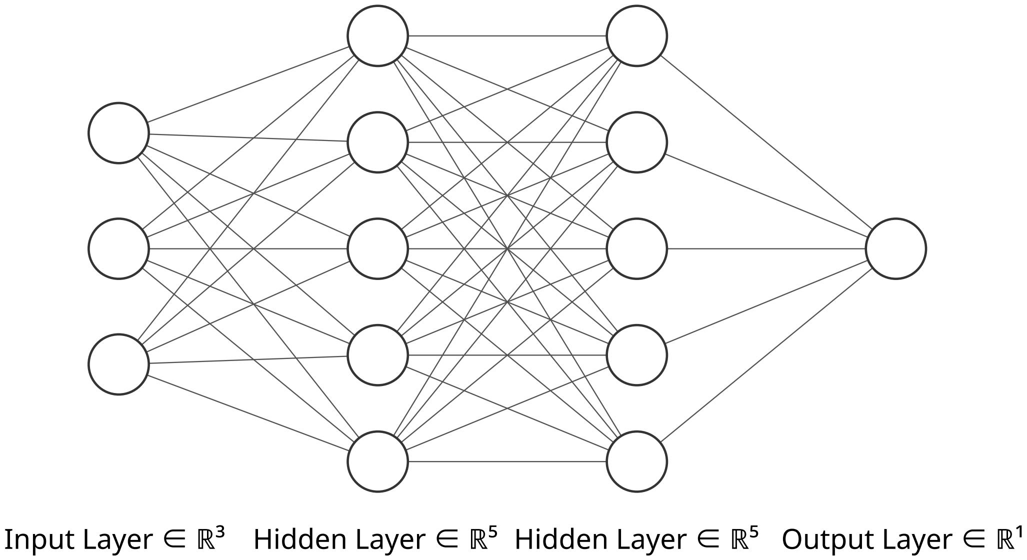


Figure 4.2: An example of a densely-connected neural network with two hidden layers. Each node on this image represents a neuron and the connections between them represents the inputs/outputs to neighbouring neurons.

4.2.2 Densely-connected neural networks

In order to build a neural network, the neurons have to be connected in some fashion. Perhaps the most straightforward method is to create layers of neurons, and then connect every neuron in a layer with every neuron in neighbouring layers. This is illustrated in Figure 4.2. Such a configuration is called a fully-connected, or densely-connected neural network. The constituents of this network structure are: the input layer where model inputs enter; hidden layers, where the bulk of model parameters live; and the output layer which outputs the model prediction(s). The training data only provides concrete, desired outputs for the overall model, and does not specify anything about the layers preceding, hence the term hidden. The neural network is free to change the parameters in these hidden layers to best approximate the target function $g(\mathbf{x})$. Output layers are functionally the same as hidden layers, except that their outputs are taken as the model prediction and so can be compared with the truth value.

The depth of a neural network is generally denoted by the number of hidden layers, or alternatively, the number of activation functions between the input and output layers. Similarly, the width of a neural network is generally referring to the number of nodes in the hidden layers.

The output of a densely-connected neural network can be formulated by repeatedly applying Eq. (4.2.1) to give

$$f(\mathbf{x};\theta) = \boldsymbol{\phi}_\theta^{(n)}(\,\ldots \boldsymbol{\phi}_\theta^{(2)}\,(\boldsymbol{\phi}_\theta^{(1)}(\mathbf{x}))\,)\,, \tag{4.2.2}$$

where $\boldsymbol{\phi}_\theta^{(n)}$ denotes the outputs and parameters of layer n, where the output layer is included in this notation. By chaining together activation functions, and especially non-linear activation functions, neural networks become good function approximators

$$f(\mathbf{x};\theta) \approx g(\mathbf{x})\,. \tag{4.2.3}$$

The universal approximation theorem states that neural networks are able to represent any continuous function, $g(\mathbf{x})$, with one hidden layer and a finite number of neurons [175]. However, it is very difficult to achieve this due to practical constraints such as limited data and network size. It is much more common to link together a larger number of hidden layers, leading to deep neural networks since better learning algorithms have been found for this case [176].

4.2.3 Loss functions and optimisation of parameters

To summarise so far, neural networks are a vast network of connected nodes with a large number of parameters that can be tuned for the problem at hand. The task of optimising the parameters is the main challenge in training a neural network and is a key area of research.

In order to quantify the performance of a network for a given task it is useful to define a loss function. For regression tasks, such as approximating a function, a commonly used loss function is the mean squared error

$$L_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(g(\mathbf{x}_i) - y(\mathbf{x}_i;\theta))^2\,, \tag{4.2.4}$$

where $g(\mathbf{x}_i)$ and $f(\mathbf{x}_i;\theta)$ are samples from the target distribution, and model predictions for inputs, $\mathbf{x}_i$, belonging to a data set with N samples. The loss function encodes the discrepancy between the truth value and the model prediction. Therefore, the task of finding an acceptable set of parameters can be reframed as an optimisation of the loss function. It should be noted that minimisation of the loss function is simply a proxy for maximising neural network predictive accuracy. Due to the training and testing datasets being finite, the generalisation of the neural network is not guaranteed for the true underlying function from which the datasets are sampled from. This problem is generally referred to as overfitting. We will refer back to this problem in the context of fitting matrix elements in Section 4.3.

Since neural networks are generally aimed at learning non-linear functions, the loss surface corresponding to Eq. (4.2.4) is a function of many variables, and is likely highly non-convex with many local minima. The methods of choice for traversing these loss surfaces are all iterative gradient-based methods with modifications to improve convergence. Broadly speaking, the algorithms iteratively update the neural network parameters based on the gradient of the loss with respect to these parameters with some step size (or learning rate) η. The update rule is given by

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta L(\mathbf{x}_{\text{batch}};\theta)\,, \tag{4.2.5}$$

where $\nabla_\theta L(\mathbf{x}_{\text{batch}};\theta)$ is the gradient of the loss with respect to the parameters of the model, averaged over the batch of inputs $\mathbf{x}_{\text{batch}}$. The initial state of the parameters are usually distributed according to normal or uniform distributions [177, 178]. This variation of gradient descent is referred to as mini-batch gradient descent because the gradient $\nabla_\theta L$ is averaged over a mini-batch of samples $\mathbf{x}_{\text{batch}}$. Mini-batch gradient descent strikes a balance between stochastic gradient descent (estimating gradient with one sample at a time), and batch gradient descent (calculating gradient with respect to entire training set) by having an efficient estimation of the gradient that is accurate enough for practical applications. In modern machine learning libraries, the updating of weights in Eq. (4.2.5) is carried out via matrix multiplications which are highly efficient on graphics-processing units (GPUs), and the gradients are computed numerically with the aid of automatic differentiation tools. For these reasons, deep neural networks have become much more widespread due to the ease of constructing and training them, as well as their good predictive performance for general problems.

The basic gradient descent algorithm described above is the basis upon which many of the most widely adopted optimisers [179–181] are based on. For an overview of these optimisers, see for instance Ref. [182].

4.2.4 Optimisation of hyperparameters

Another set of parameters that need to be optimised are the model hyperparameters. Hyperparameters are parameters that are not explicitly trainable parameters of the model, instead they control the speed and quality of the training process. For instance, the learning rate in Eq. (4.2.5) is an important hyperparameter whose initial value has to be chosen carefully to observe a good rate of convergence. Some other examples would be the number of hidden layers, the choice of activation function, or the mini-batch size.

The choice of hyperparameters is an important factor during the training process as a suboptimal set of hyperparameters can perform significantly worse than a more carefully chosen set. The process of tuning hyperparameters is often computationally expensive for two reasons. Firstly, the dimensionality of parameter space to choose from is large, meaning the number of samples required to effectively explore the parameter space grows rapidly – this is the so-called curse of dimensionality. Secondly, there is no exact a priori way to evaluate the final performance of a neural network given a set of hyperparameters, meaning training has to take place for some steps before the final model performance can be estimated.

Some of the simplest methods of hyperparameter optimisation are grid search and random search. Grid search is simply an exhaustive scan of parameters systematically sampled in a grid-like fashion, whereas random search replaces this systematic sampling with random sampling. Whilst these methods are quick to implement, the number of trials to land on a good set of hyperparameters could potentially be very large. Methods which aim to reduce the number of trials have been proposed in the literature, see Ref. [183] for a review. Below we discuss two paradigms of hyperparameter optimisation that aim to reduce number of trials when doing hyperparameter searches.

Bayesian optimisation methods have been explored to carry out hyperparameter optimisation [184, 185]. The method begins with the construction of a Bayesian probabilistic model. This model is a mapping from hyperparameter sets to the performance of a neural network, as evaluated by a loss function. This Bayesian model collects evidence through successive trainings of the neural network, and so iteratively becomes more informed on the hyperparameter space. By selecting promising hyperparameter sets based on the successively updated model, the exploration of hyperparameter space is much more efficient.

Another group of methods are based on the principle that the first few epochs of training are indicative of the final model performance [186]. This in combination with a fixed time budget gives rise to multi-armed bandit strategies [187,188] that allocate more budget to better performing models whilst simultaneously dropping poorly performing models. By iterating this procedure only the best performing models remain. Recently, there have been proposals of combining Bayesian optimisation and the bandit-based approaches [189].

In practice, the area of hyperparameter optimisation is still extremely computationally expensive even with the more advanced methods outlined above. Given a sensible set of hyperparameters, one has to balance the additional time spent tuning with the relatively incremental improvements in performance. For this reason, the hyperparameter tuning in novel machine learning experiments in high energy physics is not of paramount importance. Instead more focus is spent on the model building itself, as more sophisticated models represent a more substantial increase in performance.

4.3 Neural networks as matrix element surrogate models

Motivated by the ability of neural networks to model non-linear functions, as well as the rise in GPU usage in high energy physics, it seems natural to leverage neural networks to build surrogate models for matrix elements. There are multiple advantages to this: the time taken to evaluate matrix elements remains a bottleneck in event generation, and so building a fast surrogate model would accelerate this process. Once a fast and accurate surrogate model has been built, it can supplement traditional matrix element generators to increase statistical power of the sample by evaluating on many more phase-space points efficiently. This requires the accumulated error in the surrogate model predictions to be lower than the statistical error in the integration of matrix elements, which is a key motivation for building an accurate surrogate model.

Related to the problem of modelling matrix elements, in recent years there has been work in applying machine learning methods to learn: event weights [103], cross-sections [190, 191], analytic expressions of squared amplitudes [192], contour deformations for multi-loop integrals [193], and simplifying polylogarithms [194].

The task of building a surrogate model for matrix elements is in principle straightforward. The dataset usually consists of phase-space kinematics in the form of four-momenta and the targets are the corresponding matrix elements for the process of interest, at the desired order in perturbation theory. As already discussed in Sections 3.1 and 3.3, phase-space samplers and matrix element generators are now widely available so the acquisition of data is only limited by available computing time. Even then, this is a bottleneck only for the most expensive of processes as generation of data is trivial to parallelise. In the ideal case, the surrogate model would be faster than the matrix element generators, and sufficiently accurate. Neural network predictions are fast by virtue of them being simple matrix multiplications, and because they are predicted in batches, vectorisation is automatic[10]. Therefore the difficulty in building a good surrogate model is in controlling the accuracy.

The complication in accurately emulating matrix elements on a per-point basis arises due to the infrared divergent behaviour, as discussed in Section 2.1.4, which occurs in corners of the phase-space (UV divergences have been removed via renormalisation). Outside of these soft and collinear limits, the matrix element is a well-behaved function that varies smoothly with phase-space. However, once these infrared regions are approached, small changes in the kinematics can lead to very large changes in the matrix element. This rapid response in the target function makes it difficult to accurately model the non-divergent regions simultaneously with the divergent regions, due to the disparity in scales. This problem becomes more apparent for higher multiplicity final-states as there are more regions in phase-space for which the matrix element diverges.

Emulation of matrix elements directly have been carried out in recent years in various projects. Bishara and Montull showed [155] that it was possible to build a boosted decision tree model to reproduce the matrix elements of the loop induced process $gg \to ZZ$ with good accuracy – below 0.1% errors for fully differential distributions. However, for this $2 \to 2$ process it was already demonstrated that training a single model on the entire phase-space sampled was suboptimal compared to training a number of regressors on subdomains of the phase-space.

This trend was observed in a future work by Aylett-Bullock and Badger [4] where the authors emulated tree-level matrix elements and NLO QCD k-factors for $e^+e^- \to q\bar{q}$ + up to 3 gluons. With a higher final-state multiplicity, controlling the infrared divergences became a crucial challenge. Their approach was to first partition phase-space into divergent and non-divergent regions, then subsequently weight the divergent

[10] Event generators currently generate events one at a time, however, neural network predictions are still highly performant for single predictions.

regions further according to partition functions based on FKS subtraction. In each weighted partition, the authors trained a separate neural network, meaning that their model was an ensemble of neural networks. The performance of this method was shown to be an improvement over a single neural network trained on the unpartitioned phase-space, with good agreement in histogrammed distributions. However, the per-point agreement was lacking.

This FKS partitioning method was applied in Ref. [156] for diphoton plus jet production, where the authors presented a use case of the emulator with a novel interface to the SHERPA event generator. This work was revisited in Ref. [157], where particular attention was paid to quantifying the uncertainty associated with the neural network prediction. This uncertainty was modelled with a Bayesian neural network which enabled the boosted training on regions of phase-space that were lacking in accuracy. Additionally, the authors showed that accuracy was improved with a more careful preprocessing of the training data compared to the previous treatment. However, both of these works showed that increasing multiplicity from $2 \to 3$ to $2 \to 4$ represented a large decrease in accuracy.

To summarise, the current state of matrix element emulation has to trade-off higher multiplicity processes with accuracy. The research in this thesis attempts to tackle both of these problems by exploiting the factorisation property of matrix elements (Sections 2.2, 2.2.3, and 2.2.4), to build into the emulator the universal singular functions that describe the matrix elements in the soft and collinear limits (Sections 2.4 and 2.5). Since this method relies on the factorisation of matrix elements, we collectively refer to these family of models as factorisation-aware models. In Chapter 5, we show that by forming an ansatz out of a linear combination of Catani-Seymour dipoles with fitted coefficients, tree-level matrix elements for $e^+e^- \to q\bar{q}$ + up to 3 gluons can be emulated accurately to well below 1% per-point accuracy, even for the highest multiplicity. Extending this study to the one-loop level is done in Chapter 6, where antenna functions replace Catani-Seymour dipoles as the ingredients in the ansatz. Accuracy was observed to be of the order 1% for the 5 parton case, with good scaling to higher multiplicities.

The question of uncertainty of the neural network prediction is important as it directly contributes to the total theoretical uncertainty. An alternative to quantifying the uncertainty of the neural network prediction is to use the prediction in an intermediary step of a full calculation, where the final prediction comes from the accustomed matrix element providers. This was explored in Ref. [103], where it was shown that with a surrogate model of the event weight it was possible to use a double unweighting algorithm to increase the rate of event unweighting. In Chapter 7 we examine replacing the original emulator in that work with a factorisation-aware model to study the potential gain factors with a more accurate model. As previously mentioned, a consequence of this two-stage unweighting algorithm is that the exact event weight has to be evaluated at some point, meaning the uncertainty of the surrogate model is practically irrelevant.

Chapter 5

Emulation of tree-level e^+e^- to jets matrix elements

5.1 Motivation

The capacity of neural networks to approximate intricate functions have already been used to provide fast calculations of production cross-sections [190, 191]. In this chapter we investigate whether NNs can approximate production cross-sections more differentially by replacing computationally expensive matrix element calculations. The challenge with this endeavour is that matrix elements are plagued with numerous divergences that arise from infrared divergences. In previous works [4, 156] a combination of individual neural networks were used to approximate matrix elements. In order to deal with the complex structure of the matrix elements the authors of these studies divided the phase-space into sectors according to the infrared singularities and trained networks on these sectors, thereby limiting the complexity of the fit by isolating a single divergence per sector. All sectors were then combined to make the final prediction. While the authors of the initial study for electron-positron annihilation showed good agreement between the total cross-section and histogrammed distributions both at LO and NLO, we note that the accuracy of the interpolation at the level of individual points was a lot worse than when averaged in the histograms. In addition, the performance of the extrapolation outside of its training phase-space (i.e. more singular configurations than those considered to fit the model) is problematic.

In this chapter we present a different approach to the emulation of matrix elements that incorporates the factorisation properties of the matrix elements in the interpolation model and therefore is able to safely extrapolate the matrix element in regions more singular than that covered by the data used for the training of the emulator. We find that our interpolator also displays a much improved pointwise accuracy.

This chapter is organised as follows. Section 5.2 introduces our factorisation-aware deep neural network model, Section 5.3 showcases our results with a comparison with the model from Ref. [4] in Section 5.3.1, and an analysis of the performance of our model in Section 5.3.2. To show that the neural network is both interpolating and extrapolating well we show its behaviour on random trajectories in phase-space in Section 5.3.3. Finally, the findings of this chapter will be summarised in Section 5.4. Some discussion of specific details are collected in Appendix B so as not to distract from the main discussion.

Computer code to reproduce the methodology detailed in this chapter is provided at [195].

5.2 Fitting framework

For this work we consider the $e^+e^- \to Z/\gamma^* \to q\bar{q} + ng$ matrix elements for n up to and including 3, which corresponds to events with up to 5 jets. We denote the number of jets in the final state as n_j.

We formulate the problem of emulating matrix elements as a supervised regression task with a set of phase-space points' kinematic information as input and the values of the matrix element for each of these phase-space points as the targets. Section 5.2.3 describes how these matrix elements were obtained.

A neural network can be seen as a function $f(\mathbf{x};\theta) = \mathbf{y}$, where $f : \mathbb{R}^d \to \mathbb{R}$ maps a d-dimensional vector $\mathbf{x}$ of inputs onto the vector of outputs $\mathbf{y}$, and where θ are the parameters of the neural network which we aim to optimise such that the outputs $\mathbf{y}$ of the neural network match the target as well as possible. The simplest implementation of an emulator would be to take the input as the kinematic information of the phase-space point and the output to be the full matrix element. Our approach modifies both the input of the NN and its target, as described in the following sections.

5.2.1 Infrared divergences and dipole factorisation formula

It is well known that in soft and collinear limits the matrix element in $(n+1)$-body phase-space factorises into a singular factor and a reduced matrix element in n-body phase-space [34, 196] (Section 2.2). This factorisation was used by Catani and Seymour [51] to construct subtraction terms for the real radiation part of an NLO calculation (Section 2.4). They introduced a factorisation formula with universal dipoles that smoothly interpolates between the soft and collinear limits to capture the singular structure in these regions of phase-space. The dipole factorisation formula can be written schematically as

$$|\mathcal{M}_{n+1}|^2 \to |\mathcal{M}_n|^2 \otimes \mathbf{V}_{ij,k} \,, \tag{5.2.1}$$

where $\mathbf{V}_{ij,k}$ is a process independent, singular factor. It depends on the momenta and quantum numbers (colour and spin) of partons i, j, k, where i is the emitter parton, j is the emitted parton, and k is the spectator parton. For singly unresolved limits, this factorisation isolates all the divergent behaviour in $\mathbf{V}_{ij,k}$ and the factor $|\mathcal{M}_n|^2$ is free of divergences, which makes it more amenable to emulation through a neural network. The dipole factorisation formula forms the basis of our fitting ansatz which we present in detail in Section 5.2.2.

5.2.2 Fitting coefficients of Catani-Seymour dipoles

Instead of using a neural network to fit the matrix element directly, we use the dipole factorisation formula to build an ansatz of the colour and helicity summed $(n+1)$-body matrix element,

$$\langle|\mathcal{M}_{n+1}|^2\rangle = \sum_{\{ijk\}} C_{ijk} D_{ij,k}\,, \tag{5.2.2}$$

where $D_{ij,k} = \langle V_{ij,k}\rangle / s_{ij}$ are the spin-averaged Catani-Seymour dipoles divided by the corresponding Mandelstam invariant and C_{ijk} are the coefficients we train the neural network to fit. C_{ijk} can be interpreted as the reduced matrix element in n-body phase-space. Since the input for the C_{ijk} function is the full $(n+1)$ phase-phase information, the neural network will also model the phase-space mappings usually introduced in the factorisation formula. A schematic diagram illustrating our ansatz is given in Figure 5.1. The sum over $\{ijk\}$ denotes the sum over relevant permutations of the external outgoing legs. More detail on this is given in Section 5.2.4. The representation in Eq. (5.2.2) is not unique but through appropriate training, the neural network takes advantage of the right ingredients to model the divergent soft and collinear behaviour of the matrix elements.

This form of the ansatz allows the neural network to avoid fitting a rapidly varying function over the phase-space, leaving the Catani-Seymour dipoles to reproduce the correct singular behaviour, meaning a single neural network can interpolate a now relatively smooth function over the phase-space.

5.2.3 Data generation

For all multiplicities, phase-space is sampled uniformly using the `RAMBO` algorithm [89] with a centre-of-mass energy $\sqrt{s_{\text{com}}} = 1000$ GeV. Phase-space points are subsequently clustered using `FastJet` [197, 198] with the e^+e^- k_t algorithm [199]. Global phase-space cuts are applied according to the criterion $y_{\text{cut}} \leq y_{ij}$ where y_{ij} are the Mandelstam invariants normalised by s_{com}. Jets are clustered exclusively where d_{cut} was

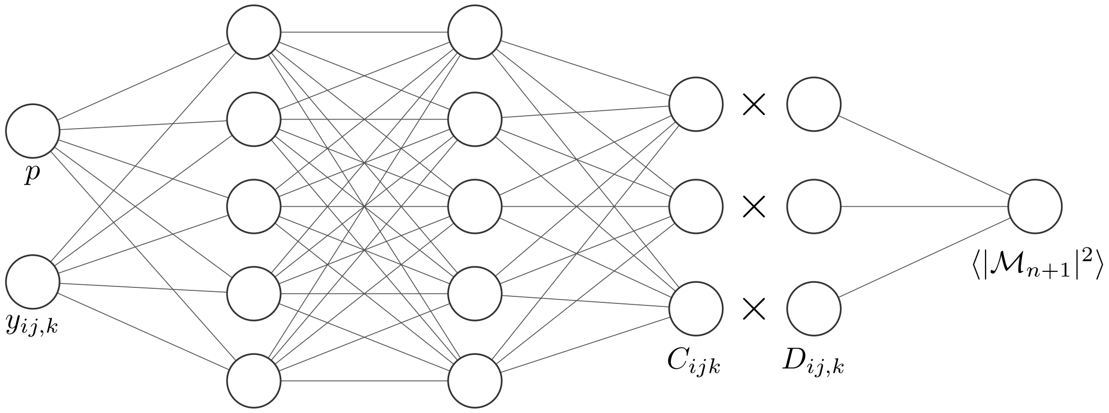


Figure 5.1: Schematic diagram of our neural network architecture. We have a densely-connected neural network with inputs phase-space points, p, and recoil factors, $y_{ij,k}$, propagated through hidden layers to the output layer which outputs C_{ijk}. These coefficients are combined with their corresponding spin-averaged dipoles as in (5.2.2) to produce an approximation of the matrix element. Diagram of neural network generated with the aid of [3].

supplied to `FastJet`. We took $d_{\text{cut}} = \max(2 \times y_{\text{cut}}, 0.01 \times s_{\text{com}})$. We explore three different values of the global phase-space cut parameter, $y_{\text{cut}} = [0.01, 0.001, 0.0001]$, to demonstrate the ability of the factorisation-aware neural network to effectively interpolate in more and more singular regions of phase-space.

The generated phase-space points are fed to the NJet package [119] to calculate colour and helicity summed tree-level matrix elements. All external legs have been considered to be massless. The strong coupling constant has been set to $\alpha_s = 0.118$[11], and the electromagnetic coupling constant has been set to $\alpha_e = 1.0/132.5070$. The mass of the Z-boson is taken to be $m_Z = 91.188$ GeV. Below we quote the full set of parameters entering the computer code used to evaluate matrix elements. We do not quote the renormalisation scale as it does not affect the value of tree-level matrix elements.

The phase-space points generated form the basis of our inputs to the neural network with the matrix elements as our fitting targets. As with most machine learning

[11] This choice of the coupling is only valid for $\mu_R^2 = m_Z^2$ and ideally, α_s should have been evaluated at the scale $\mu_R = 1000$ GeV. This change would result in slightly different plots, but does not affect the validity of the key findings of this chapter.

Parameters	
Masses	$m_b = 4.7$ GeV $m_t = 173$ GeV $m_\tau = 1.777$ GeV $m_Z = 91.188$ GeV $m_W = 80.419$ GeV $m_H = 125$ GeV
Couplings	$\alpha_s = 0.118$ $\alpha_e = 1/132.507$ $G_F = 1.166390 \times 10^{-5}$ GeV^{-2}
Decay widths	$\Gamma_t = 1.4915$ GeV $\Gamma_Z = 2.441404$ GeV $\Gamma_W = 2.0467$ GeV $\Gamma_H = 6.382339 \times 10^{-3}$ GeV

Table 5.1: List of parameters used for matrix element generation for all processes. Masses and decay widths not listed explicitly have been set to zero.

applications, we need to demonstrate that our neural network emulator has managed to generalise outside of the training dataset. We do this by firstly testing on an independent testing dataset that is never exposed to the network during training, and secondly by predicting on random trajectories in phase-space. Generation of phase-space trajectories is described in Appendix B.1. We believe that accurate predictions on random phase-space trajectories demonstrates the ability of the neural network to extrapolate to never before seen data that is of a different nature to both the training and testing datasets.

5.2.4 Neural network emulator

We construct our emulator using a densely-connected neural network built using the `Keras` API [200] with the `TensorFlow` back-end [146] with GPU support. A simple model architecture such as a densely-connected neural network allows for quicker training and inference compared to more complicated setups.

Inputs and outputs

Inputs to neural network As mentioned in Section 5.2.3, phase-space points form the basis of the inputs to our neural network. We input the 4-momenta of each outgoing parton as an $(n_j \times 4)$ array with each component of its 4-momenta standardised to zero mean and unit variance across the training dataset. Although it would be feasible to omit the energy component of the momenta or leave one outgoing parton out of the training due to our datasets being generated with a fixed centre-of-mass energy, we find that keeping all momenta information to improve the network's performance. With the acceleration of training of neural networks on GPUs, the slowdown in keeping all outgoing momenta components is negligible.

Along with the 4-momenta, we also include the recoil factors

$$y_{ij,k} = \frac{p_i p_j}{p_i p_j + p_j p_k + p_i p_k}\,, \tag{5.2.3}$$

as input for relevant permutations of $\{ijk\}$. We take the natural logarithm of $y_{ij,k}$ before standardising them to a zero mean and unit variance across the training dataset. We find that the addition of these recoil factors significantly improves performance of our neural network emulator during training and testing. This can be attributed to the fact that the coefficients C_{ijk} rely heavily on the mapped momenta in n-body phase-space, where $y_{ij,k}$ is usually required when performing the momentum mapping, see e.g. Ref. [51].

Following our ansatz in Eq. (5.2.2), we must provide spin-averaged Catani-Seymour dipoles to the neural network. These dipoles are computed for all phase-space points before training begins, but they are not passed directly to the neural network as input features. Instead, they are only included in our custom loss function which will be explained in the next subsection.

Accounting for spin-correlation in $g \to gg$ In addition to the Catani-Seymour dipoles, we include other functions to account for the spin-correlations of $g \to gg$

and $g \to q\bar{q}$ splittings which are present in the factorisation formula but averaged out in the spin-averaged dipoles. This effect becomes relevant when there are two or more gluons in the final state. We seek to capture this behaviour by introducing a pair of terms of the form

$$S_{ij} \sin\left(2\phi_{ij}\right) + C_{ij} \cos\left(2\phi_{ij}\right) \tag{5.2.4}$$

in the fitting ansatz for each gluon pair. The coefficients S_{ij} and C_{ij} are fitted by the neural network along with the dipole coefficients. The angle ϕ_{ij} is the azimuthal angle of the decay particles in the plane perpendicular to the parent particle momentum. The procedure to obtain this angle is described in Appendix B.2.

Outputs of neural network Denoting the raw output of our neural networks as c_{ijk}, they are transformed to C_{ijk} according to

$$C_{ijk} = S_{\text{coef}} \times \sinh\left(c_{ijk}\right) \tag{5.2.5}$$

where $S_{\text{coef}} = S_{\text{pred}}/S_{\text{dipole}}$. S_{pred} is the prediction scale, taken to be the minimum of the matrix elements in the training set and S_{dipole} is the dipole scale, taken to be the mean of all dipoles in the training set. These scaled coefficients are then multiplied with their corresponding Catani-Seymour dipole $D_{ij,k}$, and then summed to produce an estimation of the matrix element. The targets are the matrix elements corresponding to the phase-space point inputs in the training dataset. We transform the targets according to

$$y = \text{arcsinh}\left(\frac{\langle|\mathcal{M}_{n+1}|^2\rangle}{S_{\text{pred}}}\right), \tag{5.2.6}$$

to reduce the orders of magnitude the matrix elements span. This performs a similar transformation to taking the natural logarithm except that it remains a valid transformation for negative arguments. We require this transformation to allow negative values as arguments because the coefficients C_{ijk} will not be restricted to only positive values, meaning that during training there is a possibility for the outputs of the network to go negative. By reducing the span of the targets, the neural network is able to more effectively pick out patterns across the entire training dataset rather than a smaller region, helping it to generalise. This technique is employed in other studies, see for example [201, 202]. We would like to stress that we do not expect, and have not observed, negative predictions for the matrix element from the neural network, as this would be unphysical. y is finally standardised to a zero mean and unit variance.

Training, validation, and testing datasets In Ref. [4], the authors used 500k training samples to train their models. For a fair comparison of our respective methods, we follow their methodology closely by constructing our models to be as close as possible to theirs and generate training data by using code from their project repository [203][12]. Our model architectures will not be identical due to the difference in our methods, details of our model architecture are given in Section 5.2.4. Results of this comparison are presented in Section 5.3.1.

While training on 500k samples gives acceptable performance for the total cross-section, per-point accuracy is lacking. We find that increasing the size of the training dataset drastically improves the per-point prediction accuracy. Neural networks have been shown to scale well with large datasets [204] and given that they only need to be trained one time, it is useful to provide neural networks which have been pre-trained with maximum accuracy in mind. In addition to improving the per-point accuracy, we aim to overcome the problem of extrapolating to more singular regions. It is well known that neural networks in general do not extrapolate well [205, 206], but with our factorisation-aware model we show that letting the models learn about the infrared structure of QCD alleviates this problem.

To demonstrate the scaling performance of our model, we present our main results with models trained on more training samples where details of data generation are given in Section 5.2.3. For each multiplicity and global phase-space cut we generate

[12] For our main results we generate all data using our own code

a dataset consisting of 60 million phase-space points and their corresponding recoil factors, dipoles, and matrix elements. We then split this dataset into training, validation and testing datasets in a ratio of 4:1:1, meaning we have 40 million training samples, 10 million validation, and 10 million testing samples. The validation dataset is used to monitor model performance after each epoch of training, whereas the testing dataset is used as an out-of-sample check of the model's performance after training is complete.

Architecture

We have a fixed base to our neural network architecture that is used for all processes considered in this work, with variations in the number of nodes in the input and output layers due to the change in number of outgoing partons. Our base neural network consists of one input layer, eight hidden layers consisting of (64, 128, 256, 512, 768, 386, 128, 64) nodes, and one output layer.

The number of nodes in the input and output layers scales with the number of jets in the final state. The input layer has $(n_j \times 4) + n_{\text{rel}}$ nodes, and the output layer has $n_{\text{rel}} + (2 \times n_\phi)$ nodes, where n_{rel} is the number of relevant permutations and n_ϕ denotes the number of ϕ_{ij} angles. Relevant permutations, $\{ijk\}$, are the set of permutations for which their corresponding dipole could have the possibility to have a meaningful contribution to the matrix element for a given process. They are a subset of all the possible permutations for a given multiplicity, $P(n_j, 3)$. Relevant permutations exclude any permutation where a quark or anti-quark are emitted (i.e. $j = q$ or $\bar{q}$), as low energy quarks do not give rise to singularities in our processes of interest. Furthermore, we can remove degenerate permutations where swapping $i = g$ and $j = g$ has no effect as they have identical Catani-Seymour dipoles, e.g. $D_{34,1} = D_{43,1}$, so we only keep $D_{34,1}$. The omission of these redundant permutations speeds up training of the neural networks as we have fewer inputs, fewer dot products to compute in the loss function, as well as speeding up inference due to there being fewer dipoles to compute. For reference, we list the number of input and output

Number of final state jets	Input nodes	Output nodes
3	$(3 \times 4) + 2 = 14$	$2 + (2 \times 0) = 2$
4	$(4 \times 4) + 10 = 26$	$10 + (2 \times 1) = 12$
5	$(5 \times 4) + 27 = 47$	$27 + (2 \times 3) = 33$

Table 5.2: List of the number of input and output nodes for every process we consider.

nodes for each multiplicity we consider in Table 5.2.

The neural network weights are initialised according to the ‘Glorot uniform’ distribution as described in [177]. We use the tanh activation function for all nodes in the hidden layers, and have a linear activation function for nodes in the output layer. Initial learning rate is set to 0.001 and training mini-batch size is set to 4096. During training, we reduce the learning rate by a factor of 0.7 whenever there is no improvement in validation loss for 20 epochs. We use the `Keras` callback `ReduceLROnPlateau` to achieve this. Model training is terminated after the validation loss does not improve after 40 epochs of training using the `EarlyStopping` callback in `Keras`. We find that reducing the learning rate during training helps the model to converge to more optimal parameter sets. Since there are periods during training where the validation loss stagnates, reducing the learning rate helps to reach minima which otherwise wouldn’t be accessible due to a too large learning rate. There is a possibility to reduce the learning rate too rapidly causing the model to have suboptimal optimisation, but this is countered by the high patience we set for the `ReduceLROnPlateau` callback.

These choices of hyperparameters were not results of extensive scanning of parameter space and were chosen heuristically. Hyperparameters can be tuned more optimally using more sophisticated methods as described in Section 4.2.4, but they all rely on training a large number of models which is computationally expensive and time-consuming.

Custom loss function To assess the model’s performance when training we need to compare the network predictions with the targets. Our metric for the model’s regression performance is the mean squared error (MSE)

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - p(\mathbf{x}_i;\theta)\right)^2 , \tag{5.2.7}$$

where y_i is the target for the i-th sample out of N samples, and $p(\mathbf{x}_i;\theta)$ is the corresponding prediction obtained by combining the dipole factors and the azimuthal dependency terms multiplied by their NN-learned coefficients. In order to correct for the scale difference, we need to apply the transformation from Eq. (5.2.6) on the neural network prediction first, with the same S_{pred} that was used for scaling the targets. We choose the mean squared error to measure the network's performance because it is sensitive to outliers in the target distribution. This is useful because even though we have taken measures to reduce the span of the targets, the distribution still contains a tail towards larger values which correspond to soft and collinear configurations. These points have large contributions to the cross-section when integrating over phase-space, meaning it is important that we accurately predict these points. It is also convenient that the mean squared error tends to learn the mean of the target distribution which, in our case, corresponds to the cross-section.

In addition to the MSE, we introduce a regularisation term to penalise non-sparse representations of the matrix element. We know that in soft and collinear limits there will be dominant dipoles that have large contributions to the matrix element and there will be other dipoles with minimal contribution. We try to suppress the coefficients associated with these minimally contributing dipoles with the penalty term

$$L_{\text{pen}} = J \sum_i \frac{D_i^{-2}}{\sum_j D_j^{-2}} |C_i D_i| \tag{5.2.8}$$

where the sum over i replaces the sum over $\{ijk\}$ for brevity, and $\sum_j D_j^{-2}$ is the sum over all dipoles for a phase-space point, acting as a normalisation factor. J is a tunable parameter that scales the importance of L_{pen} versus L_{MSE}. We found that models perform the best when $L_{\text{pen}} < L_{\text{MSE}}$, so J is tuned accordingly. It is possible for the product $C_i D_i$ to be large due to D_i alone, so to penalise this we regularise the product rather than just the coefficient, since it is the product that contributes to the matrix element.

The form of L_{pen} is reminiscent of the usual L1 regularisation which promotes sparse models. Regularisation is usually included to prevent overfitting by making the model make decisions on the most important features, reducing other features to zero. In our case we would like the neural network to learn about the universal factorisation property in QCD by making it choose a minimal amount of dipoles to represent the matrix element in singular regions. In addition to preventing overfitting, Eq. (5.2.8) helps the neural network to extrapolate to more soft and collinear regions as it has learnt to choose which dipoles are relevant in specific configurations. For non-singular configurations, the neural network is free to interpolate as there is not a clear set of dipoles that dominate, meaning L_{pen} is small.

Combining the regression loss term and the regularisation term, our expression for the total loss is

$$L = L_{\text{MSE}} + L_{\text{pen}} \,, \tag{5.2.9}$$

which is minimised through mini-batch gradient descent [207] with the Adam optimiser [181] to find optimal parameters θ for the neural network.

Ensemble of models Due to the random initialisation of weights in the neural network, and the fact that the optimisation procedure is carried out on mini-batches of the full training dataset, every neural network trained will be similar but non-identical, even with identical model architecture. This is partly because the loss surface is unlikely to be a completely smooth surface with a single global minimum, instead it is likely to contain multiple local minima, meaning it would be optimistic to believe that a single neural network is able to find the most optimal set of parameters. Given that we don't expect a single network to perform optimally, we train a number of models and aggregate their predictions to create an ensemble of models. Ensembling models is a well-known technique within machine learning, for a review see [208].

Each model in the ensemble is initialised with different weights according to the 'Glorot uniform' distribution, and trained on the same but randomly shuffled dataset, resulting in models that have been exposed to different distributions of the training data. After sufficient training, each model will have a distinct set of parameters that have similar predictive power. The prediction for the matrix element is then the mean of the outputs of all models in the ensemble[13]. Taking the mean will give a more accurate and robust prediction as averaging over the different models will reduce variance due to over/underfitting in the training phase. We choose to have 20 models in our ensembles because we begin to see diminishing returns in the accuracy of per-point predictions after this. Another advantage of training an ensemble of models is that we have a measure of uncertainty, due to the neural networks, on the model predictions by calculating the standard error of the mean. That is we take the standard deviation of predictions across the ensemble and divide by the square root of number of models in the ensemble. This would not be possible with just a single model.

By choosing to build an ensemble, there is a performance impact because we have to spend more resources on training, and inference is slower due to having to predict on all models in the ensemble. However, we believe that having a more robust prediction with a measure of uncertainty outweigh these negatives. Additionally, each model only needs to be trained once, and slowdown during inference is alleviated with GPUs.

5.3 Results

In this section we present results for our matrix element emulator for e^+e^- annihilation into up to 5-jets. We first compare results obtained with our method to

[13] If not explicitly stated, all references to neural network predictions henceforth refers to the prediction from the ensemble of networks

the tree-level results of Ref. [4], then proceed to exhibit results from larger NNs that have been exposed to larger training datasets to demonstrate the full scaling performance of our model. We then further demonstrate our method's capability to generalise to unseen regions of phase-space by assessing the prediction accuracy on random phase-space trajectories that venture well outside of the phase-space region used for the training.

5.3.1 Comparison with previous work

We compare our method to methods for tree-level matrix elements emulation from Ref. [4]. The authors presented two methods: 'single' and 'ensemble' models. A 'single' model indicates that there is one neural network trained across the entire phase-space, and an 'ensemble' model indicates that there is a group of neural networks trained together with weighting functions that focus individual networks on a specific divergent region of phase-space. In order to conduct a fair comparison, we have made attempts to follow their training methodology closely by constructing the neural networks at the centre of our model with a similar structure (e.g. same hidden layer structure, same activation functions, same random initial weight distribution), training methodology (same `EarlyStopping` criteria, same initial learning rate), and have generated training datasets using code available on the authors' project repository with the relevant cuts. For the testing data, we thank the authors for providing the same testing set used in their publication.

We compare the distribution of errors for the matrix elements predictions on 3 million phase-space points. The distribution of errors is crucial because it informs us of the performance of our emulators at the matrix element level. It is well known that a neural network optimised with the mean squared error loss function has tendencies to learn the mean of the target distribution [209], meaning the quality of the cross-section prediction can potentially belie the point-to-point accuracy of emulators. In Figure 5.2, we plot the distribution of errors for matrix element predictions where our method is labelled ‘Dipole NN’. We compare against the ‘single’ and ‘ensemble’ methods by training and testing using the corresponding datasets. Note that the height of the peak for the dipole histograms are not illustrated in the figure as it would not fit on the current axes but that is not important for this comparison. We can see that the prediction-to-truth ratio distribution for our method is much narrower and consistently peaked around the ideal accuracy, indicating our model performs better on a per-point basis for all multiplicities. Even with this reduced NN size we can see that incorporating the known divergent structure explicitly in the model gives better results, as it uses the NN representation to learn a function that is more suitably approximated by a NN. For example, even though the three jet matrix element has a fairly trivial analytical structure, a standard fitting approach using a NN typically struggles to reproduce divergences. In our approach the NN only needs to emulate a non-singular modulation on top of the main divergent behaviour and is therefore more suited to the task.

5.3.2 Main results

Here we present our main results which are obtained using the larger NNs described in Section 5.2.4 along with larger training datasets described in Section 5.2.3. In Figure 5.3, we show the error distributions on 10 million matrix element predictions for each multiplicity and global phase-space cut.

Predicting on a large number of phase-space points allows us to explore singular regions with higher statistics. The ratios (right) clearly highlight the symmetry of the errors with Gaussian-like distributions tightly centred around 0. We have also included the absolute percentage difference distribution for more easily interpretable errors where the bulk of predictions are below the 0.1% error level. With increasing multiplicity, the fitting gets more challenging due to the rise in the number of singular regions in phase-space and the dimensionality of the phase-space, which can be seen in the decrease in accuracy as we increase multiplicity. Although the errors do

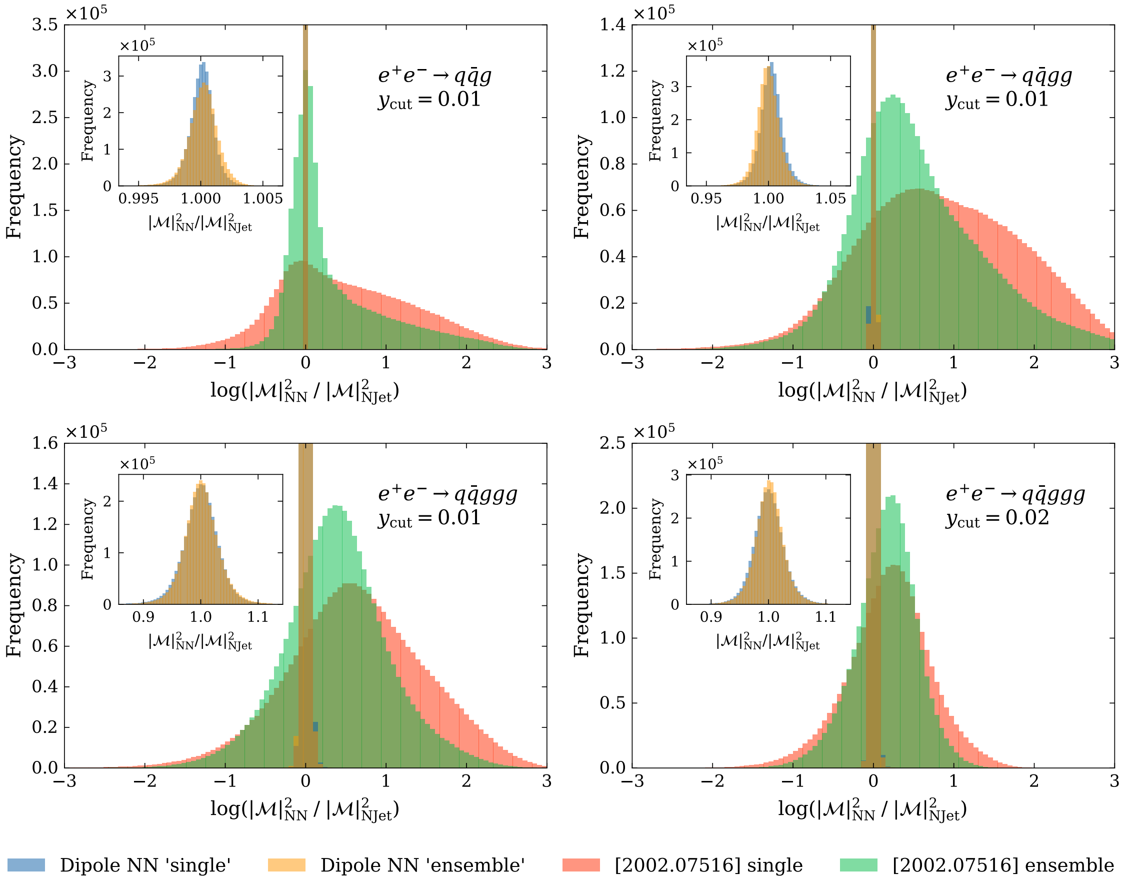


Figure 5.2: Error distribution compared to Figure 3 in Ref. [4], where data to reproduce the histograms were provided by the authors. We plot the log ratio of the matrix element as predicted by the neural network ensemble and the value from NJet on the main axes for comparison. The blue and orange dipole histograms representing our method are cut off at the top on the main axes, but the most important feature is the narrowness of the peak centred around the ideal value of 0. The insets show the detailed distribution of our result on a linear scale.

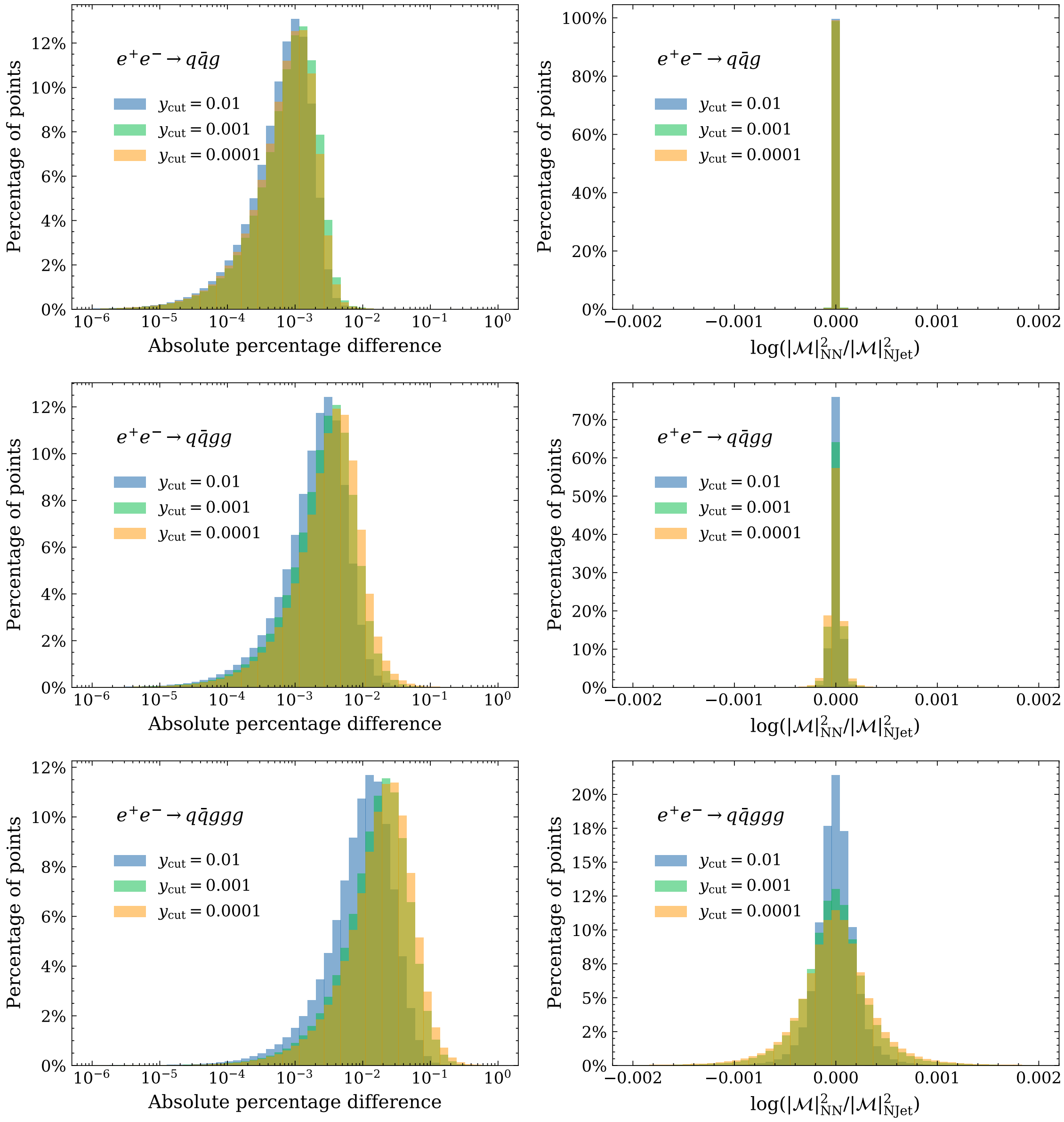


Figure 5.3: Error distributions for all three multiplicities (rows) and global phase-space cuts. Left: absolute percentage difference between NN prediction and NJet. Right: ratio of matrix elements from NN and NJet. Axis scales have been fixed for each column of subplots for ease of comparison between multiplicities.

increase, practically all matrix element predictions are below the 1% error level even for the most challenging scenario. Relaxing the global phase-space cut for the training and testing set is also expected to decrease model performance as allowing more singular regions of phase-space stretches the span of the target distribution making it difficult to fit with a single neural network. Our method manages to retain good performance while global phase-space cuts have been relaxed by a factor of 100 with only a small decrease in accuracy as illustrated in the left column of Figure 5.3. This is because most of the span is accounted for by the dipole factors, while the coefficients themselves vary less. In the 3-jet case there is negligible difference between the different phase-space cuts while the 4 and 5 jet cases see less than a factor of 10 difference in the peaks of the absolute percentage difference distributions going from $y_{\text{cut}} = 0.01$ to $y_{\text{cut}} = 0.0001$.

By increasing the size of the training datasets we aim to expose the neural network to more samples of the phase-space, thereby increasing accuracy on predictions on as much of the phase-space as possible. Along with increasing the number of training samples, the neural network architecture has been expanded to include more hidden nodes and hidden layers. The extra hidden nodes and layers introduces more parameters into the model allowing the neural network to utilise the additional data. We found that to get good performance, we had to balance the size of the training dataset used and the size of the network. i.e. a small network is not expected to capture all the variations in a large dataset as easily as a larger network would, due to the smaller number of parameters available to the network. Of course, we also had to consider more physical constraints such as the time spent on training the neural networks which limits both the size of training datasets and architecture. In Figure 5.4, we show the improvements in accuracy of our main NNs compared to the smaller NNs from Section 5.3.1. Although the training and testing data for the smaller NNs are not identical due to differences in code used to generate the sets[14],

[14] Although RAMBO is used for phase-space generation in both works, the selection criteria is different (JADE algorithm vs k_t algorithm).

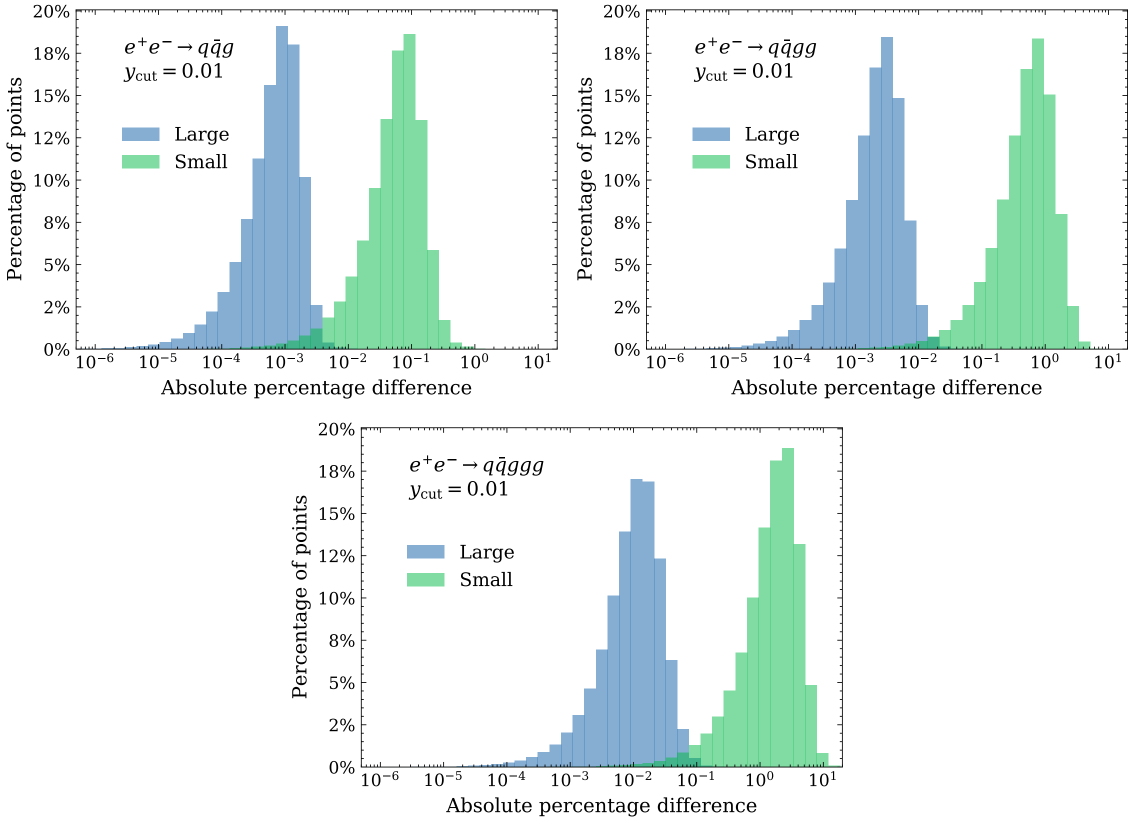


Figure 5.4: Comparison of error distributions of 10 million matrix element predictions between NNs used in Section 5.3.1 labelled as ‘Small’ and our larger NNs labelled as ‘Large’. Note that ‘small’ and ‘large’ NNs were trained on 500k and 40m training samples, respectively. The testing data is identical to those shown in Figure 5.3, for the relevant global phase-space cuts.

for this comparison it suffices to show that the larger NNs are orders of magnitude more accurate than the smaller NNs.

Improvements in per-point accuracy translate to improved total cross-section predictions. In Figure 5.5 we show the percentage differences of the NN cross-section predicted compared to those from NJet. There is a similar trend of errors increasing with increasing multiplicity and more inclusive phase-space cuts. All total cross-section predictions are well below 0.1% error. There is a small systematic offset of the neural network cross-section compared to the NJet cross-section that becomes apparent under closer inspection. This was discussed in Ref. [4] and we provide an additional explanation in Appendix B.3.

We also show in Figure 5.5 the estimated statistical Monte Carlo integration relative error for comparison. The fact that the accumulated error on the matrix element is much lower than the MC error on the cross-section opens up the possibility to use the knowledge the network has gathered on the matrix element to augment the dataset to reduce the statistical error. Using such an augmentation technique would introduce a new systematic error on the prediction related to the accumulated network interpolation/extrapolation error, which would have to be balanced with the reduction in the MC integration error. Figure 5.5 suggests that the dataset could be augmented in such a way by a large factor before reaching a minimal overall uncertainty. This opportunity might not seem very useful for this particular example of leading order matrix elements where evaluations are relatively cheap computationally, but if a similar degree of accuracy in the emulation can be obtained for higher order matrix elements, this procedure could reduce the resource cost of matrix element calculation significantly. We defer the study of this augmentation method to future work.

Since we retain good performance by relaxing the global phase-space cut, we carry out a simple test of generalisability by using the 5-jet $y_{\text{cut}} = 0.0001$ model to infer on the two datasets with harsher cuts. This is shown in Figure 5.6 where we see that accuracy is comparable to the reference (blue) in both cases. In the case of $y_{\text{cut}} = 0.01$ (left), the model trained with more of the phase-space reduces errors in the right-hand tail of the distribution. This proves that enlarging the training phase-space can be done without having a large detrimental effect on the overall accuracy, and can significantly reduce the number of large prediction errors.

5.3.3 Random trajectories

As another test of our emulator we assess the accuracy of our predictions on random phase-space trajectories. These random phase-space trajectories are generated by connecting two random points in phase-space continuously without excluding any region of phase-space. This presents an interesting and challenging test of the

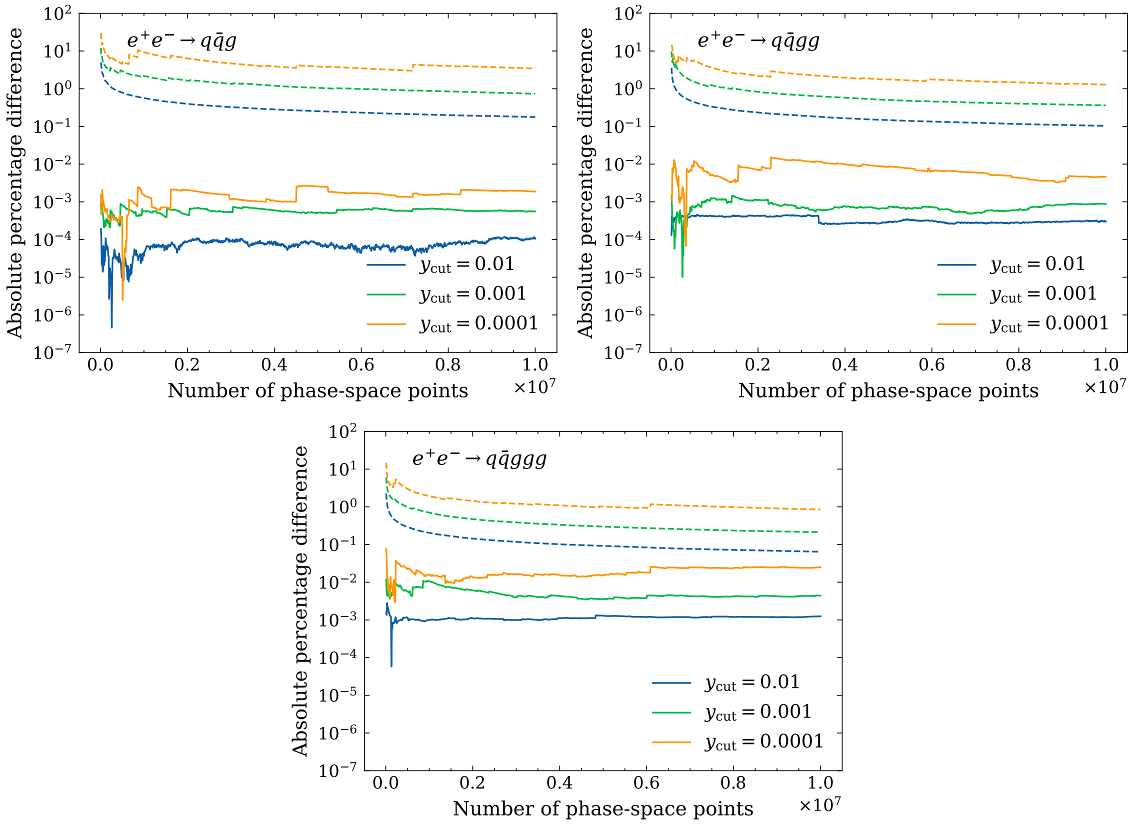


Figure 5.5: Absolute percentage differences between NN and NJet cross-section predictions in solid lines. Dashed lines represent the Monte Carlo error expressed as a percentage error relative to the NJet cross-section calculated with the corresponding number of phase-space points. Cross-sections have been calculated at intervals of 100k points up to the full 10 million phase-space points in the test set. Axis scales have been fixed to highlight the differences between multiplicities.

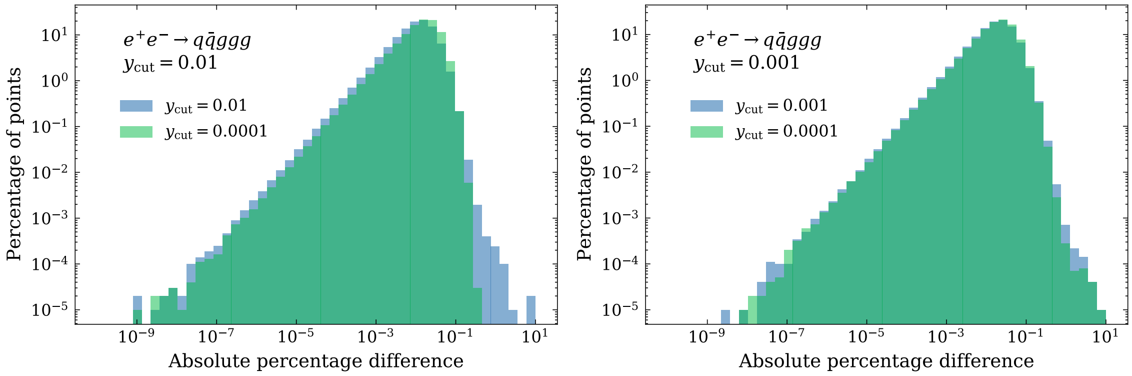


Figure 5.6: Left: error distribution on the $y_{\text{cut}} = 0.01$ testing dataset as predicted by $y_{\text{cut}} = 0.01$ and $y_{\text{cut}} = 0.0001$ models. Right: error distribution on the $y_{\text{cut}} = 0.001$ testing dataset as predicted by $y_{\text{cut}} = 0.001$ and $y_{\text{cut}} = 0.0001$ models. We use a logarithmic vertical axis to highlight the right-hand tail of the error distributions.

interpolation and extrapolation abilities of the NNs as some parts of the trajectories may lie outside of the phase-space region of the training datasets. We show the results for 5-jet trajectories as this is the highest multiplicity we considered, predictions on lower multiplicities are better-behaved. We investigated 50 different random trajectories. For the discussion in this section we chose one that contains many interesting features, namely the matrix elements span many orders of magnitudes and there are distinct peaks in the trajectory. In Figure 5.7 we show the predictions by the three 5-jet models trained on data with different global phase-space cuts for this trajectory. The left column shows the actual matrix element prediction and the right column shows the ratios of the prediction to NJet. We analysed the predictions for the 50 random trajectories and measured the fraction of their length where the accuracy falls within given intervals. Table 5.3 shows the result for the regions of phase-space where training data was available, and those falling beyond the data available to the model.

NN predictions are depicted as coloured scatter plots where the colour indicates the value of the minimum s_{ij} between any pair of final-state particles at that phase-space point. To more easily visualise the extrapolation performance of the NNs we highlight the regions where the minimum s_{ij} goes below the global phase-space cut applied to the training set the models were trained on, for each cut made. The regions of the plots where this occurs have been coloured in red. With these trajectories being completely randomly selected in phase-space there is a possibility for there to be doubly singular points or worse. To check for this we used `FastJet` to cluster the phase-space points in the same way we did for data generation, see Section 5.2.3. The pink regions indicate points which have two separate single unresolved limits, we label this configuration 'Single+single'. The purple regions indicate points which have a double unresolved limit (i.e. three particles in one jet), we label this configuration 'Double'. Although not seen in Figure 5.7, there are points which have both a double unresolved limit and a separate single unresolved limit, we label this configuration

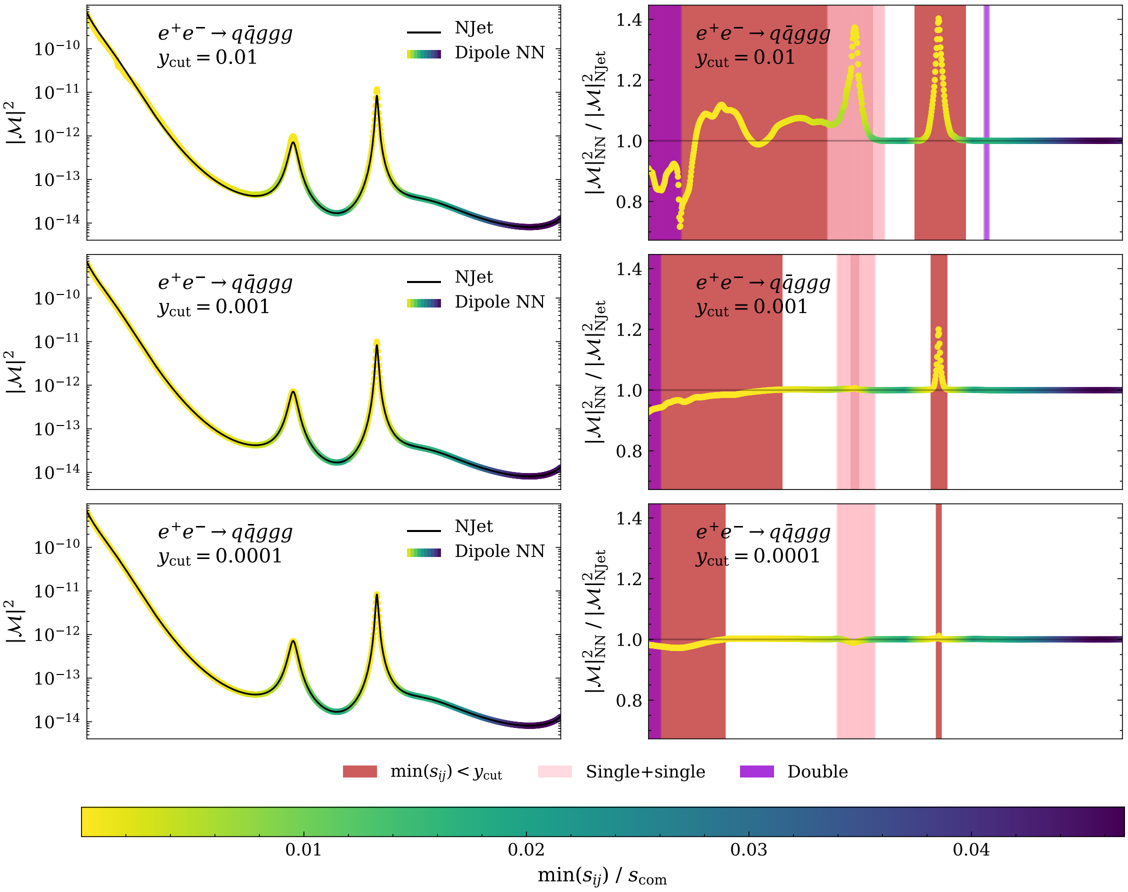


Figure 5.7: Left: matrix element prediction of random phase-space trajectory. Right: ratio of NN and NJet. NN predictions are coloured by minimum s_{ij} pair. Red bands indicate $\min(s_{ij})$ is smaller than $y_{\rm cut}$. Pink bands indicate where there are two separate single unresolved limits. Purple bands indicate double unresolved limits, i.e. three particles in one jet. The pink, and purple bands represent regions of phase-space which would have been excluded by `FastJet`.

Region	$y_{\rm cut}$	Frac. of pts	Pts outside 0.1%	Pts outside 1%	Pts outside 5%
White	0.01	36.5%	1.9%	0.0%	0.0%
	0.001	74.8%	0.98%	0.0%	0.0%
	0.0001	78.4%	1.9%	0.0%	0.0%
Pink	0.01	27.6%	58.2%	25.2%	12.9%
	0.001	13.5%	31.7%	6.5%	0.53%
	0.0001	13.5%	38.9%	9.3%	2.8%
Purple	0.01	7.2%	69.1%	32.1%	20.3%
	0.001	3.6%	31.3%	7.9%	4.5%
	0.0001	3.6%	30.5%	1.6%	0.0%
Blue	0.01	1.5%	79.3%	41.6%	0.77%
	0.001	0.4%	29.2%	0.0%	0.0%
	0.0001	0.4%	76.4%	0.0%	0.0%
Red	0.01	52.5%	76.2%	38.6%	20.5%
	0.001	7.0%	69.9%	25.2%	9.5%
	0.0001	1.1%	90.8%	33.8%	1.5%

Table 5.3: The performance of trajectory predictions separated for white, pink, purple, blue, and red regions. We present the percentage of points that lie outside 0.1%/1.0%/5.0% errors. Fraction of points indicates the percentage of points that lie in the region of interest, out of all phase-space points from the 50 random trajectories we examined.

'Double+single' and indicate is as a blue region[15]. We do not include the quark-anti-quark invariant in defining these regions as there is no associated infrared divergence. The pink, purple, and blue bands indicates regions of points which would have been discarded for our training and testing datasets.

Accuracy is high when the minimum s_{ij} of the trajectory is not below any $y_{\rm cut}$, i.e. when the NN prediction curve is blue. This is demonstrating that the NNs are interpolating well. Performance generally declines in the red, pink, purple, and blue regions, which is not unexpected as the NNs are extrapolating. Given that this trajectory has regions which go more collinear than any points the networks have been exposed to before, we would expect the networks which have been trained with the smallest $y_{\rm cut}$ parameter to perform best. We see that this is the case as accuracy is acceptable in the $y_{\rm cut} = 0.0001$ models, including in the regions where the NN is extrapolating.

[15] Another random phase-space trajectory is presented in Appendix B.4 which contains these 'double+single' regions.

In summary, we have shown that the neural networks show acceptable performance on random phase-space trajectories which are of different nature to the datasets used to train and test the networks. Given that the general performance of the NNs of all three phase-space cuts are similar, it would make sense to use the models trained with the most inclusive phase-space cuts as it has been exposed to more of the complete phase-space.

5.4 Conclusion

In this chapter we presented a new strategy to emulate matrix elements using a neural network. By leveraging the knowledge of the factorisation properties of the matrix elements our model is able to extrapolate well outside of its training range. We showed that using this method we obtain significantly improved per-point accuracy than obtained in previous works. We also showed that the per-point accuracy of the model is not significantly affected by the generation cut for the training, which means that it would be possible to train our emulator on very inclusive cuts, allowing them to be applied in a multitude of settings.

The accuracy of the emulation could allow users to augment the training dataset to reduce the MC error of cross-sections or distributions while using fewer computing resources compared to the original calculation. We leave the investigation of this aspect to further work.

Our method was demonstrated in this chapter using a tree-level process, but it could be generalised to higher order matrix elements by adapting the set of ingredients made available to the network for the interpolation. This is explicitly shown in the next chapter for the same process at NLO QCD.

Chapter 6

Emulation of e^+e^- to hadrons NLO k-factors

6.1 Motivation

In the previous chapter we described in detail the strategy for emulating tree-level e^+e^- annihilation matrix elements by exploiting the factorisation property of matrix elements. The inclusion of the dipole functions in the ansatz meant that the infrared divergent behaviour did not have to be learnt solely by the neural network itself, instead the task was reduced to learning the more well-behaved coefficients of the dipole functions.

A natural extension to the work presented in the previous chapter is to examine the emulation of the one-loop matrix elements for the same family of processes. In general, loop matrix elements are computationally much more expensive than their tree-level counterparts, meaning that the potential time saved by using a fast and accurate emulator over a more traditional one-loop provider is much higher than in the tree-level case. In this chapter we explore this avenue by using a similar procedure to Chapter 5 to construct a factorisation-aware model in order to emulate NLO QCD k-factors.

The layout for this chapter is as follows. In Section 6.2 we detail the fitting procedure. Namely, we briefly recap antenna functions, as well as describe factorisation of matrix elements within the antenna function framework in order to arrive at an ansatz of the NLO k-factor. The construction of the neural network emulator is described in Section 6.3, along with the process of training, testing and deploying the model. In Section 6.4 we present the results of the emulator demonstrating its speed and accuracy, where we see significant speed gains whilst keeping the accuracy to the 1% level. We conclude this chapter by summarising in Section 6.5.

6.2 Fitting framework for one-loop matrix element

In this chapter we consider the emulation of one-loop $e^+e^- \to Z/\gamma^* \to q\bar{q} + n_g g$ matrix elements for n_g up to and including 3, which corresponds to events with up to 5 partons. We denote the number of final-state partons as n. Instead of emulating the matrix element itself we build a surrogate for the related so-called k-factor

$$k_n = \frac{2\Re\left\{\mathcal{M}^{(n,0)}\mathcal{M}^{(n,1)\,*}\right\}}{|\mathcal{M}^{(n,0)}|^2} \equiv \frac{|\mathcal{M}^{(n,1)}|^2}{|\mathcal{M}^{(n,0)}|^2}\,, \tag{6.2.1}$$

where $\mathcal{M}^{(n,\ell)}$ denotes the amplitude for a process with n final-state partons, at loop-order ℓ. We will refer to the interference term in the numerator as the one-loop matrix element henceforth and introduce this notation for brevity. The sum/averaging over colour and helicity is implicit for both the one-loop matrix element and the tree-level matrix element. The numerator in Eq. (6.2.1) is the finite part[16] of the interference between the tree-level and one-loop level matrix element, where the conventional dimensional regularisation (CDR) scheme is used.

We choose to emulate the k-factor instead of the one-loop matrix element directly because the division of the tree-level matrix element cancels the infrared divergences

[16] See Section 3.2 and Appendix A.1 in Ref. [114] for explicit definitions.

occurring for soft and collinear external particles. However, there are still logarithmic divergences that remain from the loop integral. Another advantage is that the scale of the k-factors is naturally of the order unity, making it more amenable for emulation.

In the following, we describe how we apply the same approach as the factorisation-aware formalism introduced in Chapter 5 to encapsulate the more complex structure of the one-loop matrix element to construct an accurate emulator for the one-loop k-factors that is robust against single collinear or soft divergences.

An additional complication with one-loop matrix elements is that they are evaluated at a given renormalisation scale. This dependence can be derived from first principle, but we choose to instead incorporate this dependence into our neural network emulator as an input. This method, so-called parametric neural networks, has been utilised in other contexts [210, 211].

6.2.1 Antenna functions

In building an emulator for tree-level matrix elements in Chapter 5, Catani-Seymour dipoles [51] are sufficient to explain all single divergences arising in phase-space. For one-loop matrix elements we utilise antenna functions (see Section 2.5) which fulfil a similar purpose of providing a set of functional behaviours to build the matrix element out of.

Antenna functions as given in Ref. [52] are derived from physical colour-ordered matrix elements and by construction have the correct infrared behaviour when specific sets of particles become unresolved. For our purposes we require the set of antenna functions describing the scenario of one particle becoming unresolved at tree-level and one-loop level. Namely, following the notation of Ref. [52], these are the leading colour three-parton antenna functions X^0_{ijk} and X^1_{ijk}, respectively. X^0_{ijk} describes all configurations where parton j becomes unresolved, where i and k are the hard partons. The one-loop counterpart, X^1_{ijk}, correctly reproduces the single soft and collinear singularity structure appearing in the one-loop singular functions (see

Section 2.2.4). From this description it is clear that the antenna functions depend on the momenta in the full $(n+1)$-body phase-space.

In one antenna function, there are two hard partons which can both radiate off one unresolved parton. This is in contrast to the dipole function which only has an unresolved parton emitting from one parton. In that sense, a single antenna function is a linear combination of two dipole functions where the emitter and spectator are swapped. The advantage of this is that there are generally fewer antenna functions to consider, especially when the multiplicity increases.

Although the one-loop matrix elements data that we use for fitting are not colour-ordered, the antenna functions nevertheless provide a set of useful functions that allow the neural network emulator to form accurate approximations of the one-loop k-factor.

Since we are emulating the finite part of the one-loop matrix element we need to take care to extract all the finite parts from the one-loop antenna functions X^1_{ijk}. The full expression for the one-loop antenna we use is given as

$$X^{1,F}_{ijk} = \mathcal{F}inite(X^1_{ijk}) + \frac{11}{6}\log\left(\frac{\mu_R^2}{s_{ijk}}\right) X^0_{ijk} + \mathcal{F}(\mathbf{I}^{(1)}_{ij}(\epsilon, s_{ij}))X^0_{ijk}\,. \tag{6.2.2}$$

where the superscript F in $X^{1,F}_{ijk}$ denotes the one-loop antenna function with all finite parts extracted[17]. Most of the finite parts of the antenna function are extracted in the term $\mathcal{F}inite(X^1_{ijk})$. The second term adjusts the renormalisation scale of the antenna function from the invariant mass of the antenna partons, $s_{ijk} = s_{ij}+s_{ik}+s_{jk}$, to the renormalisation scale the one-loop matrix element is evaluated at, μ_R^2. The final term extracts the remaining finite parts from the singularity operators, where their expressions are explicitly given below for different partonic splittings. The finite part of the singularity operators can be written as

$$\mathcal{F}(\mathbf{I}^{(1)}_{ij}(\epsilon, s_{ij})) = \epsilon_0 + \epsilon_1\Re(z) + \frac{1}{2}\epsilon_2\Re(z^2)\,, \tag{6.2.3}$$

[17] Henceforth we refer to $X^{1,F}_{ijk}$ as the one-loop antenna function, unless explicitly stated otherwise.

where

$$z = \text{zlog}(\mu_R^2) - \text{zlog}(-s_{ij}) \tag{6.2.4}$$

and coefficients are given as

$$\begin{aligned}
\epsilon_0 &= \frac{\pi^2}{24}\,, \\
\epsilon_1 &= \begin{cases} -\frac{5}{6}, & \text{if } ij = qg \text{ or } gq \\ -\frac{3}{4}, & \text{if } ij = qq \\ -\frac{11}{12}, & \text{if } ij = gg \end{cases} \\
\epsilon_2 &= -\frac{1}{2}\,.
\end{aligned} \tag{6.2.5}$$

The zlog function extends the logarithm for all real-values

$$\text{zlog}(x) = \begin{cases} \log(x), & \text{if } x \geq 0 \\ \log(|x|) - i\pi, & \text{otherwise}\,. \end{cases} \tag{6.2.6}$$

6.2.2 Factorisation of matrix elements

In the following, we recap the factorisation of matrix elements as seen in Sections 2.2 and 2.2.4 within the antenna function framework.

Tree-level matrix elements in $(n+1)$-body phase-space can be factorised in the single soft and collinear limits as

$$|\mathcal{M}^{(n+1,0)}|^2 \longrightarrow X_{ijk}^0 |\mathcal{M}^{(n,0)}|^2\,, \tag{6.2.7}$$

where $|\mathcal{M}^{(n,0)}|^2$ is the reduced matrix element in n-body phase-space and X_{ijk}^0 is the three-parton tree-level antenna function introduced in Section 2.5. The one-loop matrix element similarly exhibits factorisation in the soft and collinear limits. This has been extensively studied [53, 55, 56, 58, 59] with the splitting kernels computed [52, 53, 66]. Schematically, in the single soft and collinear limits, the one-loop matrix

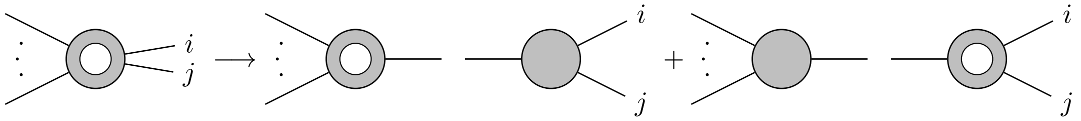


Figure 6.1: Diagram illustrating factorisation of one-loop matrix element. In a singly unresolved limit, the $(n+1)$-body one-loop matrix element tends to the sum of a n-body one-loop matrix element multiplied by a tree-level splitting kernel and an n-body tree-level matrix element multiplied by a one-loop splitting kernel. The tree-level elements are drawn as a fully filled in circle, while the one-loop elements are drawn as a donut shape.

element can be deconstructed into

$$|\mathcal{M}^{(n+1,1)}|^2 \longrightarrow X^0_{ijk}|\mathcal{M}^{(n,1)}|^2 + X^{1,F}_{ijk}|\mathcal{M}^{(n,0)}|^2\,, \tag{6.2.8}$$

where $X^{1,F}_{ijk}$ is the three-parton one-loop antenna function. This equation can be thought of as a tree-level splitting kernel multiplied by a one-loop reduced matrix element, plus a one-loop splitting kernel multiplied by a tree-level reduced matrix element. This is illustrated pictorially in Figure 6.1.

6.2.3 Ansatz for the k-factor

Given that both the tree-level and one-loop level matrix element factorise in the soft and collinear limits, we can rewrite the $(n+1)$-body k-factor in these limits as

$$\begin{aligned} k_{n+1} &\longrightarrow \frac{X^0_{ijk}|\mathcal{M}^{(n,1)}|^2 + X^{1,F}_{ijk}|\mathcal{M}^{(n,0)}|^2}{X^0_{ijk}|\mathcal{M}^{(n,0)}|^2} \\ k_{n+1} &\longrightarrow \frac{|\mathcal{M}^{(n,1)}|^2}{|\mathcal{M}^{(n,0)}|^2} + \frac{X^{1,F}_{ijk}}{X^0_{ijk}} \\ k_{n+1} &\longrightarrow k_n + \frac{X^{1,F}_{ijk}}{X^0_{ijk}}\,, \end{aligned} \tag{6.2.9}$$

where we see that the k-factor tends to a sum of the reduced k-factor, k_n, and a ratio of antenna functions. By summing over ratios of antenna functions for all limits of a given process, we can construct an ansatz for the k-factor over all of phase-space.

n	Tree-level antenna	One-loop antenna	{i,j,k} permutations
3 ($q\bar{q}g$)	$A_3^0(q,g,\bar{q})$	$A_3^1(q,g,\bar{q})$	(1, 3, 2)
4 ($q\bar{q}gg$)	$A_3^0(q,g,\bar{q})$ $D_3^0(q,g,g)$	$A_3^1(q,g,\bar{q})$ $D_3^1(q,g,g)$	(1, 3, 2), (1, 4, 2) (1, 3, 4), (2, 3, 4)
5 ($q\bar{q}ggg$)	$A_3^0(q,g,\bar{q})$ $D_3^0(q,g,g)$ $F_3^0(g,g,g)$	$A_3^1(q,g,\bar{q})$ $D_3^1(q,g,g)$ $F_3^1(g,g,g)$	(1, 3, 2), (1, 4, 2), (1, 5, 2) (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5), (3, 4, 5)

Table 6.1: List of antenna functions we use for each process, and the full list of {i,j,k} permutations, where $q = 1$, $\bar{q} = 2$, $g = 3, 4, 5$.

This informs our ansatz for the k-factor, which is given as

$$k_{n+1} = C_0 + \sum_{\{ijk\}} C_{ijk} \frac{X_{ijk}^{1,F}}{X_{ijk}^{0}} \tag{6.2.10}$$

where C_0 and C_{ijk} are coefficients fitted by the neural network. C_0 is an additive term aiming to model the reduced k-factor, and C_{ijk} are multiplying the ratio of antenna functions to fit the single collinear, and soft limits in multiple regions of phase-space. The sum over $\{ijk\}$ denotes the sum over the relevant permutations of final-state partons, accounting for all the relevant singular limits of the k-factor. This sum allows the neural network to make use of all the provided antenna functions to make an approximation of the colour-summed matrix element. The full set of antenna functions which we implement into our model is detailed below.

The antenna functions are generically written as X_{ijk}^{ℓ}, which in practice is replaced with specific antennae containing either $qg\bar{q}$, qgg ($\bar{q}gg$), or ggg, which are referred to as A, D, and F antennae, respectively. The antennae listed in Table 6.1 are sufficient to describe all infrared singularities in the partonic processes we consider.

Since the k-factor has infrared divergences arising from unresolved partons in the final-state being removed, and with the appropriate antenna functions accounting for the logarithmic divergences from the loop momenta, the challenging task of fitting a rapidly varying function over phase-space is reduced to fitting a group of well-behaved coefficients that dictate how to suitably utilise the antenna functions.

6.3 Building the neural network emulator

Dataset generation

Phase-space is sampled uniformly using the `RAMBO` algorithm [89] with a centre-of-mass energy $\sqrt{s_{\text{com}}} = 1000$ GeV. The global phase-space generation cut is set to $y_{\text{cut}} = 0.0001$. We have shown in [5] that the accuracy of the emulation is not greatly affected by the generation cut, but the extrapolation performance is increased with a more inclusive cut, so we have chosen this value. `FastJet` [197, 198] is used to exclusively ($d_{\text{cut}} = 0.01 \times s_{\text{com}}$) cluster final-state jets with the e^+e^- k_t algorithm such that there is at most a single unresolved parton.

For each phase-space point generated, we sample a renormalisation scale, $\log(\mu_R)$, from a uniform distribution with end points at $[\log(\sqrt{s_{\text{com}}}/4), \log(4\sqrt{s_{\text{com}}})]$. In other words, we sample the renormalisation scale logarithmically. We observe that the neural network manages to learn the renormalisation scale dependence well therefore opt to sample μ_R in a wider range than is usually used for the conventional scale variations which varies μ_R up and down by factors 2.

Generated phase-space points are fed to MadGraph [114, 115] to compute the tree-level and one-loop level matrix elements. For each phase-space point, we use the corresponding renormalisation scale sampled and evaluate the strong coupling constant at this scale using the NNPDF-4.0 NNLO PDF set [30] with the LHAPDF6 interface [26]. The full set of parameters used to evaluate the matrix elements are quoted in Table 6.2.

We generate 1100k data points in total, using 100k points for training and validation, leaving 1 million points for independent evaluation of model performance. Training on a limited dataset is a realistic scenario for processes which are prohibitively expensive, and we show that it is possible to build an accurate emulator with the relatively small number of data points. Note that because we are sampling μ_R along with phase-space simultaneously, there is an extra dimension in the sampled

Parameters	
Scales	μ_R sampled logarithmically in range $[\log(\sqrt{s_{\text{com}}}/4),\, \log(4\sqrt{s_{\text{com}}})]$
Masses	$m_b = 4.7$ GeV $m_t = 173$ GeV $m_\tau = 1.777$ GeV $m_Z = 91.188$ GeV $m_W = 80.419$ GeV $m_H = 125$ GeV
Couplings	$\alpha_s = \alpha_s(\mu_R^2)$ $\alpha_e = 1/132.507$ $G_F = 1.166390 \times 10^{-5}$ GeV^{-2}
Decay widths	$\Gamma_t = 1.4915$ GeV $\Gamma_Z = 2.441404$ GeV $\Gamma_W = 2.0467$ GeV $\Gamma_H = 6.382339 \times 10^{-3}$ GeV

Table 6.2: List of parameters used for matrix element generation for all processes. α_s is evaluated with the NNPDF-4.0 NNLO PDF set. Masses and decay widths not listed explicitly have been set to zero.

space. This means that the 100k training points we have are not comparable to 100k training points if we had not sampled over μ_R. In practice, we have found that sampling over μ_R has a small impact on accuracy but opt to go this route to have the flexibility to predict over a range of μ_R.

Inputs to model

As inputs to the neural network we provide the 4-momenta of all final-state partons. The renormalisation scale enters the network as $\log(\mu_R)$ as we expect the dependence on μ_R to be in the form of a logarithm. Following Ref. [5] we include the phase-space mapping variables to aid the network in learning the reduced matrix element information. Namely, we include as inputs r and ρ from Ref. [53]

$$r_{ijk} = \frac{s_{jk}}{s_{ij} + s_{jk}}\,, \tag{6.3.1}$$

$$\rho_{ijk} = \sqrt{1 + \frac{4r(1-r)s_{ij}s_{jk}}{s_{ijk}s_{ik}}}\,, \tag{6.3.2}$$

where i and k are the hard radiating partons, and j is the unresolved parton. We include the subscript $_{ijk}$ on r and ρ to represent the explicit dependence on the specific set of momenta used to calculate them. To improve training, we transform these variables as $r \to \log(r)$ and $\rho \to \log(\rho - (1 - \varepsilon))$, where ε is a small constant added to improve the numerical stability. It is taken to be $\varepsilon = 10^{-8}$. An additional input that we have observed to increase accuracy of the emulator are the Mandelstam invariants. These are fed into the model as $\log(s_{ij})$ for all pairs of final-state particles. All inputs are standardised to zero mean and unit variance.

Outputs of model

The outputs of the neural network are the fitted coefficients C_0 and C_{ijk} in Eq. (6.2.10), which when combined with the antenna functions produces an approximation of the k-factor. We then recover the one-loop matrix element by multiplying by the corresponding tree-level matrix element. We do not provide the tree-level matrix element in the emulator as the evaluation time is generally much lower than that of the one-loop matrix element, and it is usual for one-loop matrix elements to be evaluated at phase-space point sets where the tree-level matrix element has already been unweighted, so that only the k-factor is required.

To train the network we compare the target k-factors from MadGraph to the predictions from the neural network in the loss function by providing the network with the antenna functions. The target distribution is standardised to zero mean and unit variance. Since the k-factors are of order unity we do not need to do any additional pre-processing to aid the network in training.

Neural network architecture

A schematic of the neural network model is given in Figure 6.2. We build the neural network emulator with `Keras` [200] and `TensorFlow` [146]. The emulator is a dense neural network with three hidden layers of 64 nodes each. This network size was

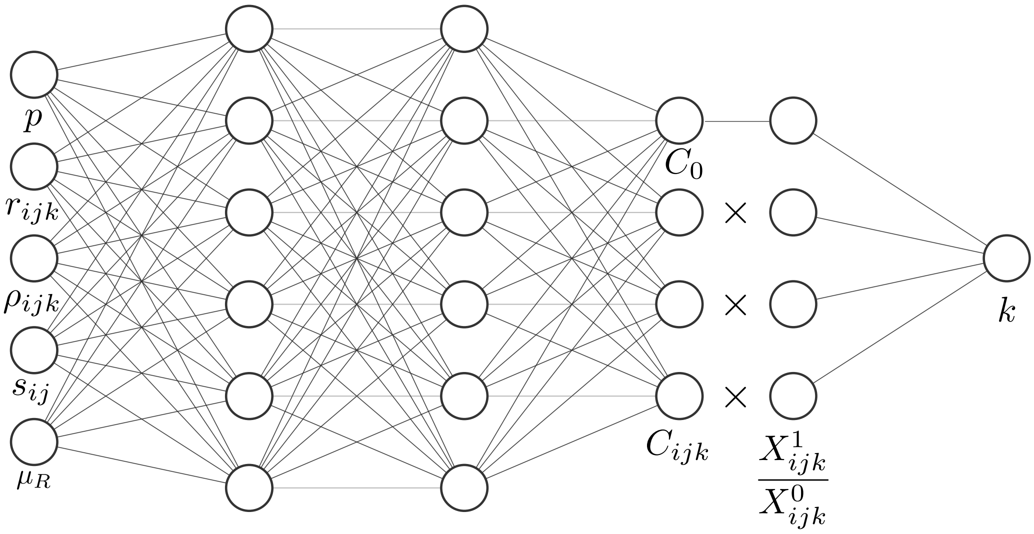


Figure 6.2: A schematic diagram of the neural network emulator. The emulator is a dense neural network with inputs: phase-space points, p, momenta mapping variables, r_{ijk} and ρ_{ijk}, kinematic invariants, s_{ij}, and renormalisation scale μ_R. The outputs of the network are the fitted coefficients, C_0 and C_{ijk}, as given in Eq. (6.2.10).

chosen with consideration given to the number of training samples and to reduce the discrepancy between training and validation loss (i.e. a larger network is more prone to overfitting on the training set if there is insufficient data to fit the additional weights). For the remaining hyperparameters we summarise the network architecture in Table 6.3.

We choose to use the mean absolute error (MAE) as the loss function for training because it is precisely the error measure we would like to minimise. The error for one prediction is given as

$$k_{\text{true}} - k_{\text{pred}} = \frac{|\mathcal{M}^{(n,1)}|^2_{\text{true}} - |\mathcal{M}^{(n,1)}|^2_{\text{pred}}}{|\mathcal{M}^{(n,0)}|^2_{\text{true}}} = \Delta\,, \tag{6.3.3}$$

where the error in the one-loop matrix element normalised by the tree-level matrix element is what we want the neural network to minimise. Since k-factors are a ratio of matrix elements, the numerical values it can take are not unique for a given value of the tree-level matrix element and/or one-loop matrix element. For example, for two similar values of the k-factor, the scales of the matrix elements going into each ratio may be vastly different. To ensure that the network remains accurate for large

Table 6.3: Hyperparameters of the neural network and their values.

Parameter	Value
Hidden layers	3
Nodes in hidden layers	[64, 64, 64]
Activation function	swish [173, 174]
Weight initialiser	Glorot uniform [177]
Loss function	MAE (k-factor), MSE (one-loop matrix element)
Batch size	256
Optimiser	Adam [181]
Learning rate	10^{-3}

Callbacks	`EarlyStopping`, `RatioEarlyStopping`
	`ReduceLROnPlateau`

values of the tree-level matrix element, where corresponding corrections contribute more to the total cross-section, we weight the training points by

$$w_i = \log\left(\frac{|\mathcal{M}^{(n,0)}|_i^2}{\min(|\mathcal{M}^{(n,0)}|_i^2)}\right) , \tag{6.3.4}$$

where the i index denotes training samples. Although we use the MAE on the k-factors as the training loss, we terminate model training based on the one-loop matrix element accuracy. This is done by monitoring the mean squared error (MSE) between the model prediction and corresponding truth value at the end of each training epoch. This takes advantage of the compact k-factor distributions for training purposes, but bases model selection on the accuracy for the physical one-loop matrix elements.

To reduce the effects of overfitting we have two `EarlyStopping` criteria: the first is to stop training once the validation loss has not improved in 100 epochs, and the second is a `RatioEarlyStopping` which terminates training if the ratio of training loss to validation loss drops below a certain threshold. We take this threshold to be 0.9. We also use the `ReduceLROnPlateau` callback as a way to adapt the learning rate during training. The learning rate is reduced by a factor of 0.7 whenever validation loss plateaus with a patience of 20 epochs. We find that with these hyperparameters we achieve a balance of reducing overfitting and quick training times. On average the models train in approximately 20 minutes on an Nvidia P100 GPU.

Although we build and train our model using TensorFlow, we deploy the model using the Open Neural Network Exchange (ONNX) runtime [212] with the CUDA execution provider to run predictions on an Nvidia P100 GPU. With the optimised operations in the ONNX runtime, we see that compared to TensorFlow the model inference time is reduced by an order of magnitude or more. Another advantage is that it gives users the flexibility to move the pipeline away from TensorFlow on Python to a more generic interface to the neural network model. One example would be to integrate the ONNX model into a C++ workflow for use with current event generators to replace the one-loop provider with a neural network emulator.

In addition to the CUDA execution provider, we will use the ONNX runtime CPU execution provider to compare with MadGraph for a comparison of single CPU core performance. This will be the closest to a real world benchmark as event generators typically generate events on a single core.

Although we find that the neural network models converge well, to account for stochasticity in the training, the random initialisation of model parameters, and to reduce variance on predictions, we initialise 20 models for training and use the mean of these models as our model prediction. This ensembling will also give a measure of the uncertainty due to the neural network optimisation, by using the standard deviation of the 20 replica model predictions.

We plot in Figure 6.3 the losses of the 20 replica models for the 5 jet process, where we have plotted the loss for the k-factor (MAE) and the one-loop matrix element (MSE). We can see that the models have all converged to a similar point when training is terminated. The noise in the validation loss at the beginning of training can be attributed to the fitting of the coefficients: when the model is learning how to pick the relevant combination of antenna functions there can be large variations in the prediction, however, the variations become much smaller once the model learns the factorisation properties and converges. We can see the variations in the validation loss are small at the end of training, and that there is not a large discrepancy between the training and validation losses, as enforced by the RatioEarlyStopping callback.

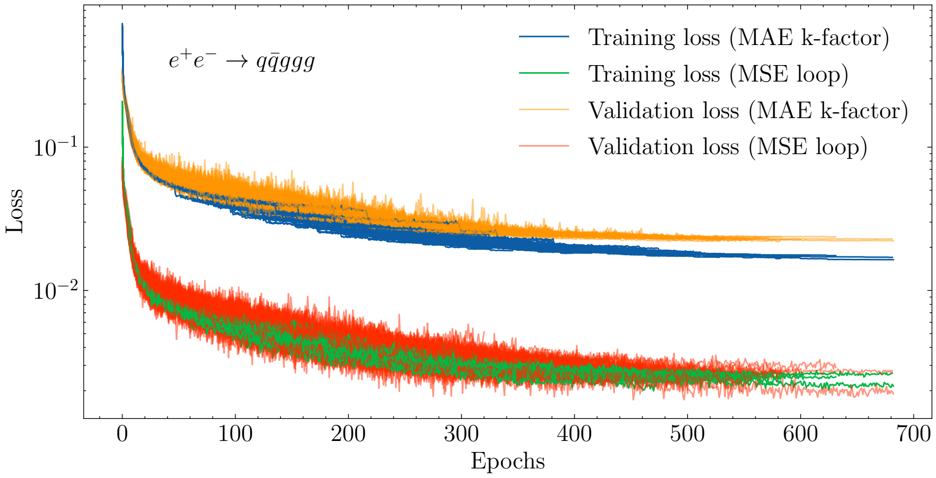


Figure 6.3: Training and validation losses of the 20 independent initialisations of the ensemble replicas plotted as individual curves for the 5 parton model. We plot the MAE loss for the k-factor and the MSE loss for the one-loop matrix element. The scale difference between the MAE and MSE losses is a consequence of the form of the loss functions and is not surprising. We see that the replica models all converge to a similar point with the validation loss being close to the training loss when training is terminated. The step feature in the training loss is due to the `ReduceLROnPlateau` callback.

6.4 Results

6.4.1 Error distributions

In this section we present results for our NLO QCD k-factor emulator for e^+e^- annihilation into up to 5-jets. First we show a comparison between our model described in Section 6.3 which we label ‘antenna’, and a ‘naive’ model with no factorisation properties built into the emulator: a densely connected neural network with the parameters given in Table 6.3 that directly predicts the k-factor. For the ‘naive’ models we train with the MAE loss on the k-factors with no modifications and terminate training based on this loss. As with the ‘antenna’ models, we also ensemble 20 individual replica models for the ‘naive’ model predictions. In Figure 6.4 we compare histograms of Δ for all final-state multiplicities between these two models. It is immediately clear that building in the factorisation structure of the matrix elements greatly increases accuracy, with increasing relative improvements for the higher multiplicity cases. The ‘antenna’ error distributions are symmetric,

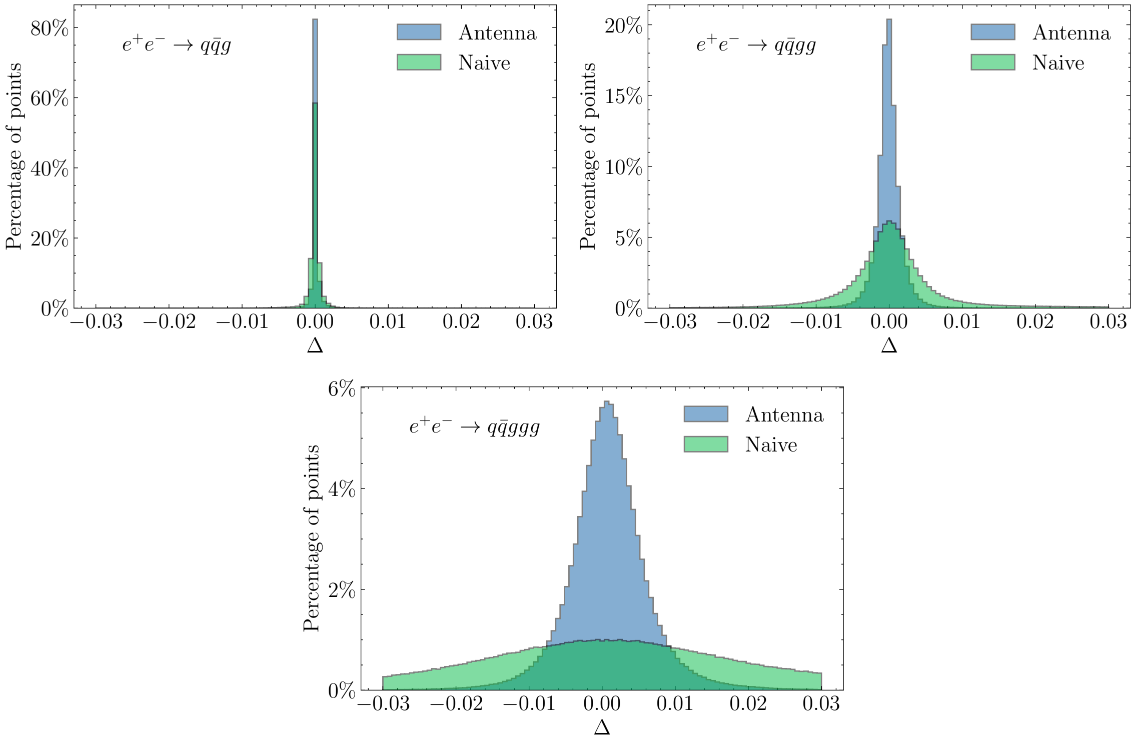


Figure 6.4: Error distribution in terms of Δ for all multiplicities. ‘Antenna’ model is as described in this article, and ‘naive’ model is a simple densely connected neural network model without any factorisation properties built in. We keep the horizontal axis scale fixed for all subplots to make it easier to compare accuracy across the different multiplicities.

strongly peaked around the ideal value of 0, and with tails falling off rapidly. We see the general trend of increasing multiplicity decreases accuracy, however we observe that the bulk of the 5 jet final-state is within percent accuracy and with the lower multiplicities well below this.

To show that we are accurate across the entire span of the tree-level matrix elements, and to have a closer inspection of the tails of the Δ distribution, we plot a 2d histogram of Δ against the value of $|\mathcal{M}^{(5,0)}|^2$ in Figure 6.5. The bulk of phase-space points are contained in the high population bins depicted in yellow, representing the peaked distribution seen in Figure 6.4, whereas the green to purple coloured bins represent the tails of the low population Δ distribution. We see that the accuracy stays contained inside a band and does not flare out as the magnitude of the tree-level matrix element increases. This shows that we manage to fit the k-factor even in the infrared and collinear limits where the tree-level matrix element becomes large. On

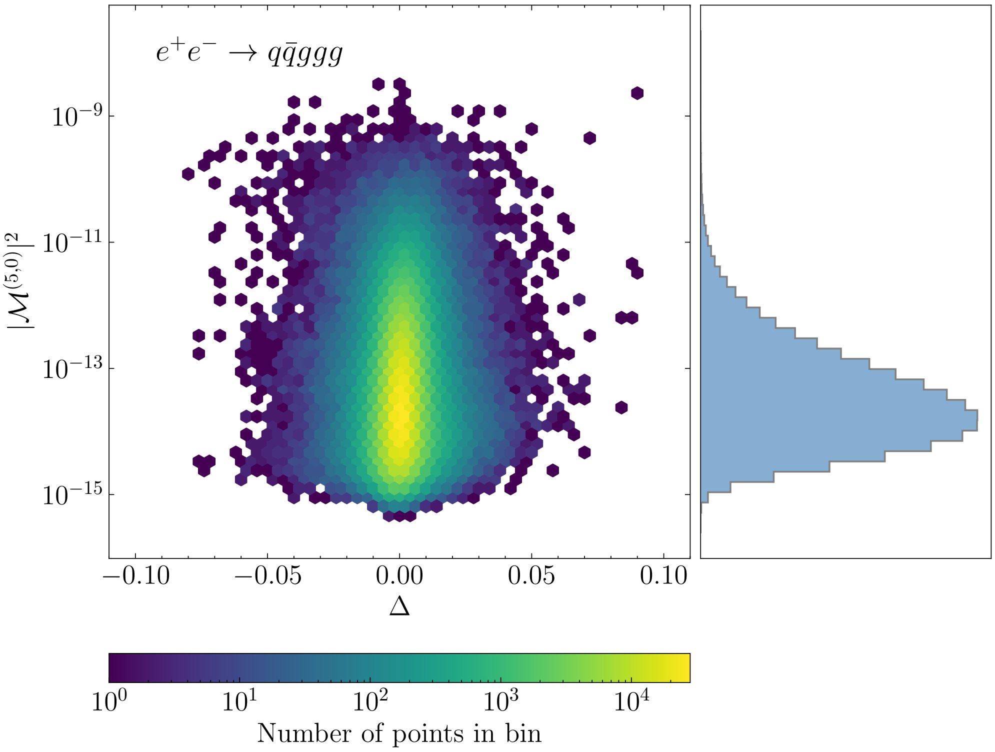


Figure 6.5: Left: Δ error distribution plotted against the tree-level matrix element. Yellow bins indicates high density regions of points and purple bins indicates single points. Right: marginal distribution of tree-level matrix elements. This illustrates that the network is able to reproduce a good approximation across all sampled phase-space.

the right-hand side subplot we plot the distribution of the tree-level matrix elements where we can see that that even with relatively few training points in the tails, the emulator is still able to predict these regions as well as where there is more abundant data.

6.4.2 Renormalisation scale dependence

In Figure 6.6 we show that our emulator has learned the renormalisation scale dependence, independent of the antenna functions. To produce a trajectory we first sample a phase-space point with the same cuts as described in Section 6.3, then we evaluate the k-factor at this phase-space point with μ_R varying from $\sqrt{s_{\text{com}}}/8$ to $8\sqrt{s_{\text{com}}}$. We choose to sample from a wider range than used for training to examine the μ_R extrapolation performance. After subtracting the sum of the antenna

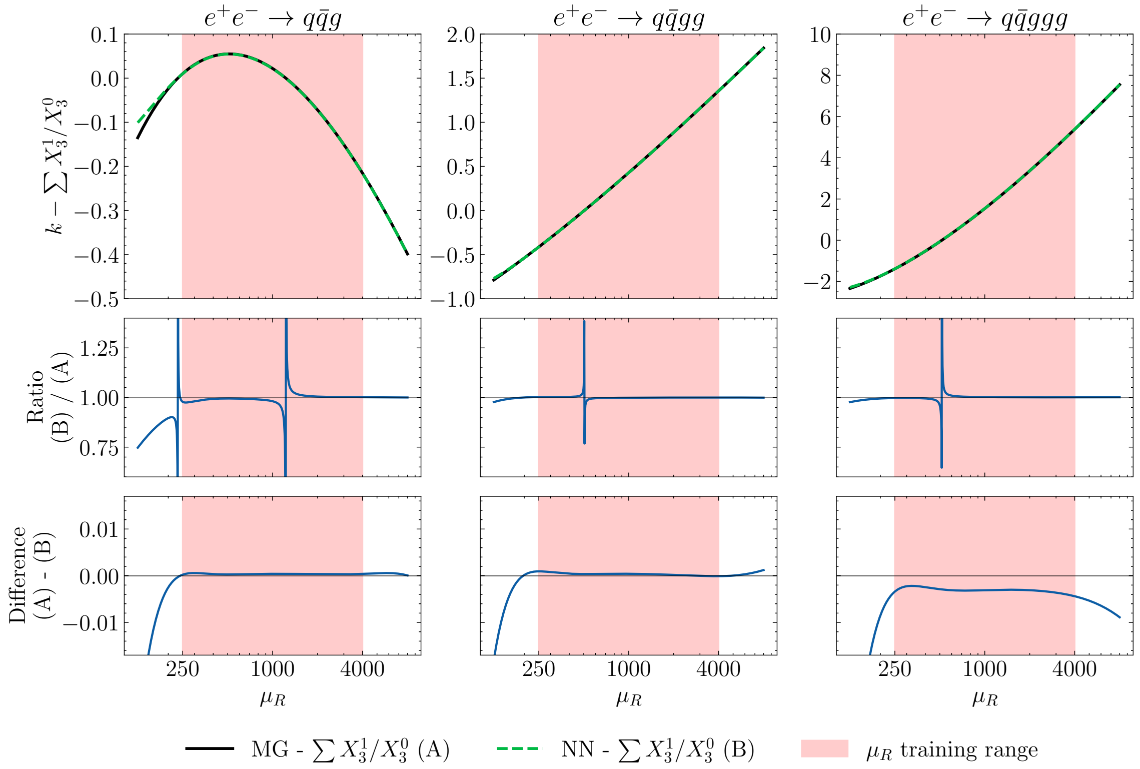


Figure 6.6: Renormalisation scale trajectories for all multiplicities. These trajectories are predictions of the k-factor with the sum of antenna functions subtracted to show renormalisation scale dependence in the remainder. Each trajectory is at one phase-space point sampled from `RAMBO`, with μ_R spanning the range $[\sqrt{s_{\text{com}}}/8, 8\sqrt{s_{\text{com}}}]$. Each trajectory is composed of 1000 points. The region that μ_R was uniformly sampled from for training is indicated as a red band. The error bands on the NN predictions are too small to be seen.

functions we see that the remainder still has a dependence on the renormalisation scale that is accurately captured by the neural network. As with the Δ distribution plots in Figure 6.4, there is a slight decrease in accuracy as we increase the multiplicity, however, the ratios and differences are well-behaved throughout the entirety of the trajectories inside the range of training data. The only anomaly occurs when the trajectories cross zero, causing spikes in the ratios, but the difference in truth and prediction remains well-behaved around these regions. Outside of the training range we see an acceptable extrapolation, but given that the training range is wider than the range in which the renormalisation scale is normally varied for scale variations, we do not find this problematic.

6.4.3 Total cross-section predictions

In Figure 6.7 we show that we reproduce the total cross-sections over an integration of 1M phase-space points, for all multiplicities, at three different values of the renormalisation scale representing the nominal value ($\mu_R = \sqrt{s_{\text{com}}} = 1000$ GeV) and the two variations usually used to estimate scale uncertainties. Note that these phase-space points were generated independently of those in Figure 6.4, and that we are integrating the tree-level and one-loop interference, not the k-factor. We multiply our NLO k-factor prediction with the MadGraph tree-level matrix element to reproduce the loop-matrix element for integration. The neural network errors are well below the statistical Monte Carlo integration error.

By neural network error we are referring to the absolute percentage difference to the cross-section, and not the errors due to neural network optimisation. For that, we examine the variations in the 20 replica model predictions in Figure 6.8 where we plot the total cross-section predictions as a scatter plot. The blue band illustrated is one standard deviation of the 20 predictions made. We see that the true value of the total cross-section is within the one standard deviation band. Not shown in the figure is the Monte Carlo statistical error which as seen in Figure 6.7 dominates the

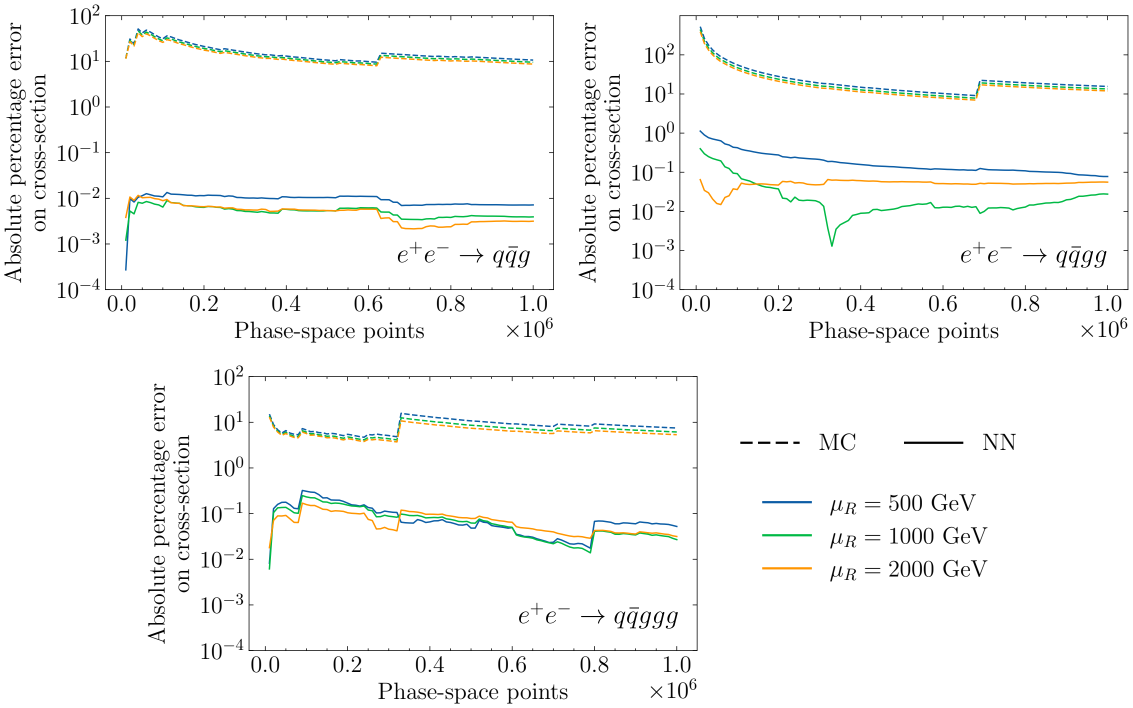


Figure 6.7: Error on the total cross-section across an integration of 1 million phase-space points. For each multiplicity, we evaluate the matrix elements at the three values of the renormalisation scale as reported. The solid lines are the absolute percentage error in the true total cross-section and the neural network predicted value. The dashed line represents the statistical Monte Carlo integration error which falls as $1/\sqrt{N}$. The jumps in error are due to large values of the matrix element being integrated.

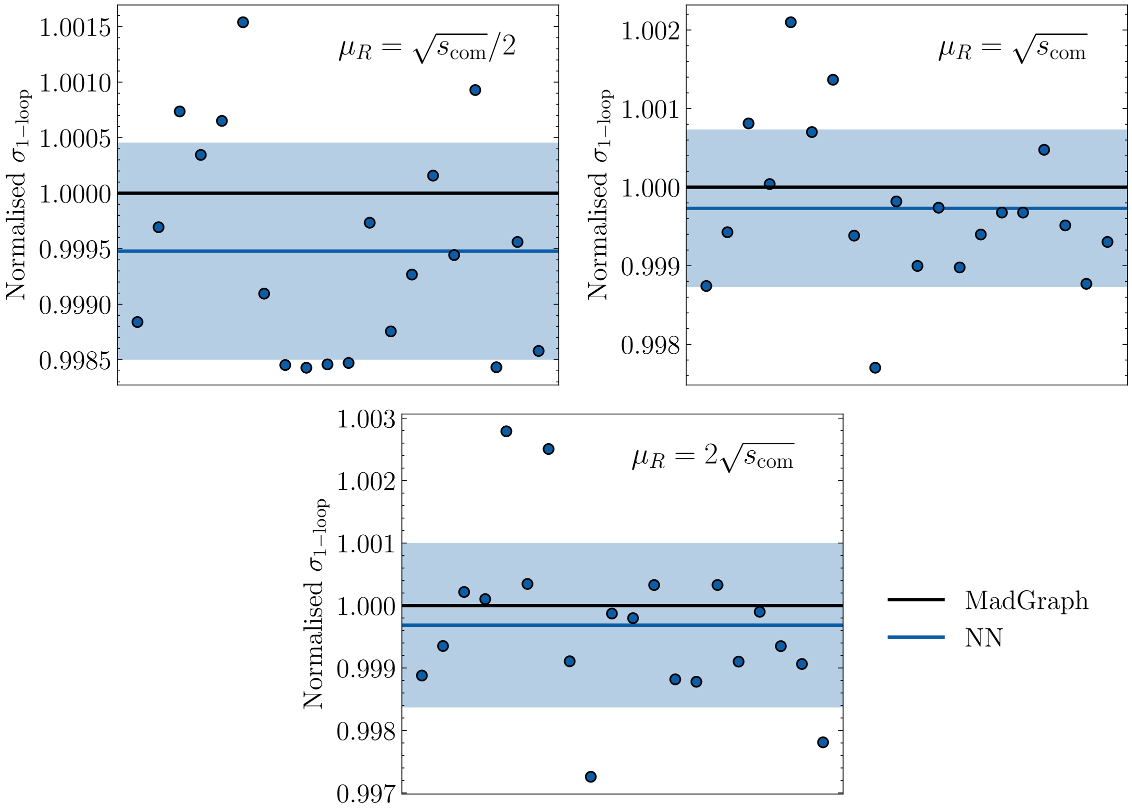


Figure 6.8: Scatter plot showing the variation in predictions of the $e^+e^- \to q\bar{q}ggg$ total cross-section for the 20 replica models, where the plotted cross-section is normalised by the truth value provided by MadGraph. Total cross-sections in each subplot are evaluated at the renormalisation scale quoted in the upper right. The mean prediction of the 20 replica models is drawn as the horizontal blue line with one standard deviation illustrated by the blue band. The total cross-section as predicted by MadGraph is plotted as the horizontal black line.

absolute error between the NN predictions and the true value.

Since the discrepancy between the true total cross-section and NN predicted total cross-section is so small, this strongly indicates that once we fit on the relatively small training set, we can predict with good confidence on many more phase-space points to get a prediction of the cross-section before reaching the same level of error as the Monte Carlo integration error. This can also be seen from the shape of the NN error, it is relatively constant once enough points have been integrated.

6.4.4 Model evaluation time

In Figure 6.9, we plot the evaluation time of the emulator for both the CPU and `CUDA` (GPU) execution providers in the `ONNX` runtime, as well as the reference time from MadGraph. For the NN (GPU) predictions we predict on 1M phase-space points concurrently, whereas for the NN (CPU) predictions, we predict on one point at a time. The times reported are then the mean of the total evaluation times. We see that compared to MadGraph, our NN emulator is faster for all multiplicities, with the advantage being largest for the 5 jet case, with speed gains of over four orders of magnitude when utilising GPU acceleration. The advantage of using the GPU is not only from being able to batch process the predictions, it is also to leverage the auto-vectorisation tools provided by `TensorFlow` to accelerate the model input computations.

While the evaluation time of the GPU accelerated NNs are by far the quickest, and would be the ideal scenario for a NN emulator to be used, event generation typically occurs on CPUs where phase-space points are evaluated one at a time. This is precisely why our NN (CPU) predictions were made on single phase-points and not over batches, to showcase what the performance would be like when embedded in a typical production workflow. We observe that even with this constraint, the NN emulator is much quicker in the higher multiplicity cases.

One of the main bottlenecks in the NN (CPU) prediction is using the NN ensemble to infer on single phase-space points, this is illustrated by the weak scaling in multiplicity in the left subplot of Figure 6.9 and is illustrated explicitly in Figure 6.10. Since the model architecture is identical for all multiplicities other than the final output layer, there will not be much difference in cost. Another bottleneck is the computation of model inputs which contain large, complex expressions with many evaluations of logarithms and dilogarithms. In our `Python` implementation, the computation time of these inputs is comparable to the model inference time in the 5 jet case, as shown in Figure 6.10. Time taken to compute model input scales with final-state partons

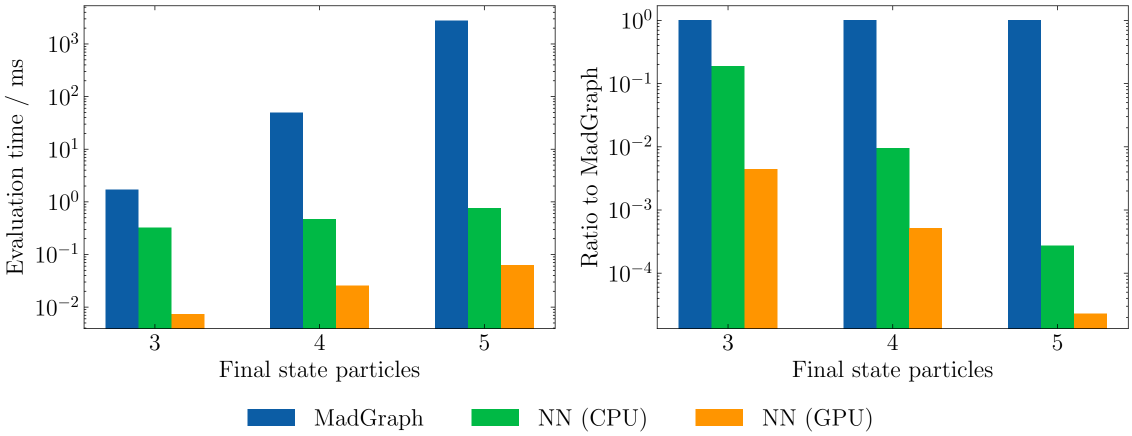


Figure 6.9: Left: evaluation time in milliseconds of a single phase-space point. Times quoted for MadGraph are averaged over 1000 random phase-space points. Times quoted for NN are averaged over 1M random phase-space points. Right: ratio of evaluation times to MadGraph. GPU used is Nvidia P100 16GB, and CPU is Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz.

because of the increase in number of antenna functions in the ansatz to account for the larger number of singular configurations, as well as a larger number of the other input variables.

By moving away from a `Python` oriented workflow to a `C++` workflow, we anticipate the model inputs can be computed much more quickly, potentially reducing the total prediction time by a factor of 2 for the 5 jet case. However, the model prediction time is already highly optimised by the `ONNX` runtime, so we do not expect much improvement to be possible on that front.

For the NN (GPU) predictions, the model inference time is negligible compared to the model input computations since we take advantage of predicting on the entire batch of 1M phase-space points at once, and so once averaged across this dataset each single point takes an insignificant amount of time. The model input computations are vectorised on the GPU and so we see an order of magnitude reduction in time taken to calculate them compared to computing them on a single core of a CPU.

We note that our emulator retains good performance even when we go to higher multiplicity. In Ref. [157] the authors expressed concern about the fact that the accuracy of the surrogate model is decreasing with the multiplicity of the process. We

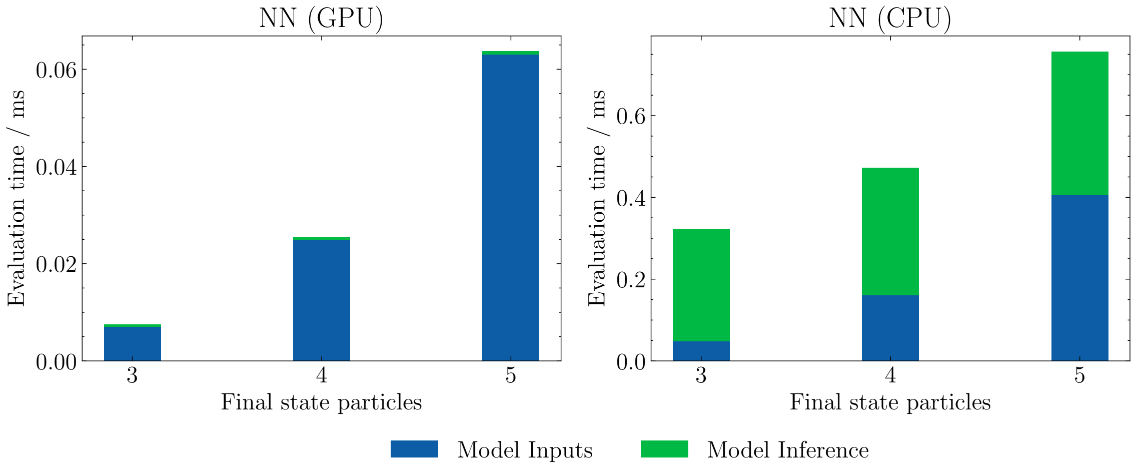


Figure 6.10: Breakdown of the total time taken to predict on a phase-space point (averaged over 1M predictions) into model input computations and the actual model inference for both GPU and CPU model deployments. The model inference portions in the NN (GPU) subplot are very small.

show that by incorporating suitable physically motivated functions in the ansatz that the network accuracy drops off much less rapidly when going to higher multiplicities.

6.5 Conclusion

In this chapter we presented the extension of Chapter 5, where we introduced the factorisation-aware model. By adapting the ingredients provided to the neural network model to a more suitable set, we have been able to move from the emulation of tree-level matrix element to NLO k-factors. We have shown that the philosophy of incorporating relevant physics information into the model greatly improves accuracy of predictions even for matrix elements with a more complex divergent structure.

The results presented demonstrate that predictions are at the percent level for the most demanding process, with accuracy in the infrared regions of phase-space being well-behaved. We have also shown that for relatively few training points, the model is able to learn the target function well enough to evaluate many more phase-space points before the accumulated error matches the statistical integration error. Furthermore, the uncertainty of cross-section predictions associated with optimisation of model parameters was shown to contain the truth value. This provides evidence that the neural network model can be used to augment existing samples with additional phase-space points with confidence given that the absolute error of the cross-section prediction is relatively constant.

In addition to being accurate, we have given evidence that the model, although optimally deployed on a GPU, is orders of magnitude quicker than traditional loop providers on a single CPU. By deploying the model with the ONNX runtime a generic interface to the neural network model is available in many programming languages, allowing it to be embedded into modern event generators which are mainly written in C++.

Chapter 7

Emulation of hadron-hadron initiated matrix elements

7.1 Motivation

In the previous two chapters, we detailed the emulation of e^+e^- annihilation tree-level matrix elements and NLO QCD k-factors. We have shown that in e^+e^- annihilation processes we are able to construct emulators that accurately reproduce the matrix elements, whilst keeping the evaluation time much lower than existing matrix element providers.

The Large Electron Positron collider (LEP) was the last high energy e^+e^- collider to be in operation with data collection terminating in 2000 to prepare for the LHC. Lepton-lepton colliders planned for the future [213] are not scheduled to begin operating for at least another decade. Therefore, we turn our focus to researching techniques to make predictions more efficient for the LHC, which will be collecting data until the end of the 2030s, namely, adapting the factorisation-aware model for hadronic collisions. To showcase the extension to hadronic collisions, we consider several partonic channels contributing to $Z+4$ and 5 jets production, and $t\bar{t}+3$ and 4 jets at leading order. These high-multiplicity processes are of particular importance for physics at the LHC as they contribute to the background of BSM searches. Furthermore, these processes typically have very low unweighting efficiencies.

In Ref. [103] a novel two-stage unweighting procedure was introduced that used a lightweight neural network as a surrogate model for the full event weight. This procedure utilises the rapid evaluations of the surrogate model to first unweight a trial event, then correct for the mismatch by re-weighting with the full event weight, where this second re-weighting step has a much higher efficiency. The authors showed that it was possible to accelerate event unweighting with this procedure but found that for some processes this procedure was slower than the standard one-step rejection sampling (see Section 3.2). In this chapter we explore whether a more accurate emulation of matrix elements via the use of the factorisation-aware neural network model would lead to further acceleration of event unweighting and realise an improvement for the cases where it did not.

The layout of this chapter is as follows. In Section 7.2 we detail the extension of the model applied in Chapters 5 and 6 to hadronic collisions. The novel two-step unweighting procedure introduced in Ref. [103] is recapped in Section 7.3. The application of the factorisation-aware model in this two-step procedure, and implementation in the SHERPA event generator framework will be discussed in Section 7.3.1. In Section 7.4 we will showcase the emulator accuracy and display the acceleration in the generation of unweighted events. Finally, we will conclude the chapter in Section 7.5 by summarising our findings.

7.2 Neural network emulator framework

7.2.1 Extension of ansatz to hadronic collisions

The methodology introduced in Chapter 5 for emulating e^+e^- annihilation matrix elements can be straightforwardly adapted to hadronic collisions by accounting for the radiation coming from the initial-state partons. This is achieved by extending the set of dipole functions in the ansatz of the colour and helicity summed $(n+1)$-body matrix element. The ansatz is a sum over all permutations of dipole functions relevant for a given channel, each coming with a corresponding coefficient. It is given by

$$|\mathcal{M}_{n+1}|^2 = \sum_{\{ijk\}} C_{ijk} D_{ij,k} \,, \tag{7.2.1}$$

where emitter and spectator partons, denoted by i and/or k, can now be initial- and final-state partons. Therefore, the set of dipoles D_{ijk} now includes the final-final (FF), final-initial (FI), initial-final (IF), and initial-initial (II) dipoles, as opposed to just the FF dipoles for e^+e^- annihilation. The coefficients C_{ijk}, which are more well-behaved than the matrix element in the soft and collinear limits, are fitted by the neural network and can be interpreted as reduced matrix elements. These coefficients are the outputs of the neural network, which forms an approximation of the matrix element once combined with the appropriate dipole functions.

An additional extension that we study in this chapter is the inclusion of massive dipoles in the ansatz of the matrix element. This enables the examination of pure QCD processes with massive particles which is of particular importance for top quark pair production with jets. To that end, we include the massive FF, FI, and IF dipoles into the ansatz in Eq. (7.2.1). The massive dipole functions are generalisations of their massless counterparts, meaning it would be possible to only use the massive dipole functions for simplicity. However, only using the massive dipoles when a dipole contains a massive parton saves on computational costs as the massless dipole functions are cheaper to evaluate.

To showcase these extensions, we emulate a selection of partonic channels at tree-level for $Z+4$ and 5 jets, and $t\bar{t}+3$ and 4 jets production at the LHC. We summarise the partonic channels considered in Table 7.1. Together, these processes allow us to examine the performance of the extensions detailed above. We note that with all massless and massive dipoles implemented into the modelling framework, it is in principle possible to take advantage of the factorisation-aware model to cover

Process	Partonic channel(s)
$Z+4j$	$gg \to e^- e^+ g g d \bar{d}$
$Z+5j$	$gg \to e^- e^+ g g g d \bar{d}$
$t\bar{t}+3j$	$u\bar{u} \to t\bar{t} g d \bar{d}$ $gg \to t\bar{t} g g g$
$t\bar{t}+4j$	$ug \to t\bar{t} g g g u$

Table 7.1: List of partonic channels considered in this chapter.

QCD-enhanced behaviour at tree-level.

7.2.2 Generation of data

Data is generated using the SHERPA event generator with the AMEGIC matrix element generator. Below we list the selection criteria for the phase-space sampling and document the parameters used for matrix element evaluation in Table 7.2.

Z+jets

For Z boson production in association with jets, we consider the partonic channels $gg \to e^- e^+ g g d \bar{d}$ and $gg \to e^- e^+ g g g d \bar{d}$ at leading order, representing tree-level contributions to $Z + 4$ jets and $Z + 5$ jets production at the LHC, respectively. The phase-space sampled for generating training and testing data is constrained by requiring a dilepton invariant mass $m_{e^- e^+} > 66\,\text{GeV}$ and four or five anti-k_t jets [214] with radius parameter $R = 0.4$ and $p_{T,j} > 20\,\text{GeV}$. We consider a proton-proton centre-of mass energy of $\sqrt{s} = 13\,\text{TeV}$ and use the NNPDF-3.0 NNLO PDF set [215].

$t\bar{t}$+jets

For top quark pair production, we consider three partonic channels that contribute to $t\bar{t} + 3$ jets and $t\bar{t} + 4$ jets in proton-proton collisions. These are pure QCD processes with massive particles which pose a challenge due to the top quarks carrying colour charge, meaning there is a significant proliferation of Feynman diagrams when

Parameters	
Scales	$\mu_R = \mu_F = 91.188$ GeV for Z+jets $\mu_R = \mu_F = 175$ GeV for $t\bar{t}$+jets
Masses	$m_d = 0.01$ GeV $m_u = 0.005$ GeV $m_s = 0.2$ GeV $m_c = 1.42$ GeV $m_b = 4.8$ GeV $m_t = 173.21$ GeV $m_e = 0.511$ MeV $m_\mu = 0.105$ GeV $m_\tau = 1.777$ GeV $m_Z = 91.1876$ GeV $m_W = 80.385$ GeV $m_H = 125$ GeV
Couplings	α_s set by PDF $\alpha_e = 1/137.03599976$
Decay widths	$\Gamma_t = 2$ GeV $\Gamma_\tau = 2.26735 \times 10^{-12}$ GeV $\Gamma_Z = 2.4952$ GeV $\Gamma_W = 2.085$ GeV $\Gamma_H = 4.07 \times 10^{-3}$ GeV

Table 7.2: List of parameters for the matrix element generation of all partonic channels. α_s is set by the NNPDF-3.0 NNLO PDF set. Masses and decay widths not listed explicitly have been set to zero.

considering their jet-associated production. For the processes contributing to $t\bar{t} + 3$ jets we require three anti-k_t jets with $R = 0.4$ and $p_{T,j} > 20\,\text{GeV}$. The phase-space of the process $ug \to t\bar{t}gggu$ contributing to $t\bar{t}+4$ jets is constrained by requiring four staggered anti-k_t jets with $R = 0.4$, $p_{T,1} > 100\,\text{GeV}$, $p_{T,2} > 50\,\text{GeV}$, $p_{T,3} > 40\,\text{GeV}$, and $p_{T,4} > 20\,\text{GeV}$. We do not impose phase-space cuts for the external top quarks. They are treated as on-shell in the matrix element calculation, $p_t^2 = p_{\bar{t}}^2 = m_t^2$ with $m_t = 173.4\,\text{GeV}$, and only decayed a posteriori.

7.2.3 Constructing the emulator

The emulator is a dense neural network built using the `Keras` API to the `TensorFlow` backend, and is deployed using the `ONNX` runtime. The network consists of four

Table 7.3: Summary of hyperparameters for the neural network employed to emulate matrix elements for all partonic channels in Table 7.1.

Parameter	Value
Hidden layers	4
Nodes in hidden layers	128
Activation function	swish [173, 174]
Weight initialiser	Glorot uniform [177]
Loss function	MSE
Batch size	512
Optimiser	Adam [181]
Initial learning rate	10^{-3}
Callbacks	`EarlyStopping`, `ReduceLROnPlateau`

hidden layers, each containing 128 nodes. We use the swish activation function for all layers, and initialise node weights according to the Glorot uniform distribution. We use the data generated from the SHERPA event generator (described above in Section 7.2.2) to fit the neural network by minimising the mean squared error. Training is optimised with the Adam optimiser with an initial learning rate of 10^{-3}. Learning rate is reduced by a factor of 0.7 during training when validation loss shows no improvement for 30 epochs, and model training is terminated once there is no improvement in the validation loss after 60 epochs. We summarise the neural network hyperparameters in Table 7.3.

As inputs to the neural network, we feed in: the 4-momenta of initial- and final-state particles, the phase-space mapping variables for each class of dipole function[18], and the kinematic invariants s_{ij} for all pairs of particles in the process. We refer to all phase-space mapping variables as $y_{ij,k}$ for brevity, following the notation of Eq. (7.2.1). To aid the network in training, we preprocess the phase-space mapping

[18] The mapping variables for FF, FI, IF, II dipoles are $y_{ij,k}$ (Eq. (2.4.4)), $x_{ij,a}$ (Eq. (2.4.13)), $x_{ik,a}$ (Eq. (2.4.19)), and $x_{i,ab}$ (Eq. (2.4.23)), respectively.

variables in the following manner[19]

$$y_{ij,k} \to \begin{cases} \log(y_{ij,k}) & \text{if massive FI dipole}\,, \\ \log(1 - y_{ij,k}) & \text{if massless FI, IF, or II dipole}\,, \\ \log(y_{ij,k}) & \text{otherwise (massless FF dipole)}\,, \end{cases} \tag{7.2.2}$$

such that their distributions have similar shape and width. The kinematic invariants are also transformed with the logarithm as they can span many orders of magnitude. All inputs are standardised to zero mean and unit variance, with each component of the 4-momenta being standardised separately.

The target matrix elements are preprocessed by the transformation

$$|\mathcal{M}_{n+1}|^2 \to \operatorname{arcsinh}\left(\frac{|\mathcal{M}_{n+1}|^2}{S_{\text{pred}}}\right)\,, \tag{7.2.3}$$

where S_{pred} is the prediction scale taken to be the minimum matrix element in the training set. The raw outputs of the neural network, c_{ijk}, are transformed to the coefficients appearing in the ansatz via the transformation

$$C_{ijk} = S_{\text{coef}} \times \sinh(c_{ijk}) \tag{7.2.4}$$

where S_{coef} is the coefficient scale, defined as $S_{\text{pred}}/S_{\text{dipole}}$. The dipole scale, S_{dipole}, is the representative value of a dipole, which we take to be the median of all dipoles in the training set. A schematic diagram of the neural network is given in Figure 7.1.

In Chapters 5 and 6, the final predictions made for the matrix elements were the mean of an ensemble of independent replica models. Here we train 10 replica models to monitor the convergence of training, however, we select only the model with lowest validation loss for predictions. As a demonstration of model convergence, the training and validation loss of 10 replica models for the $gg \to e^-e^+gg d\bar{d}$ channel is shown in Figure 7.2. This convergent behaviour is also seen in the other partonic channels.

[19] Since we do not have top quarks in the initial-state, we do not mention massive IF, or II dipoles in Eq. (7.2.2).

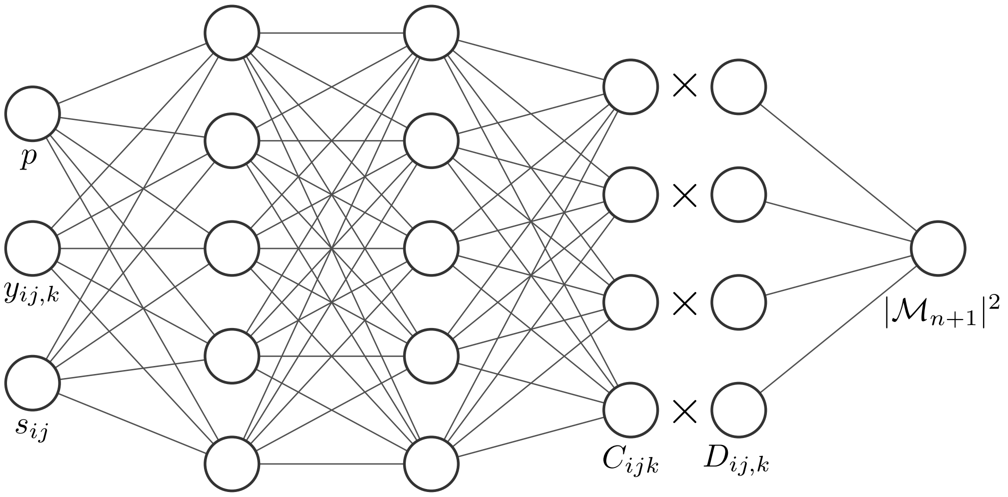


Figure 7.1: Schematic diagram of neural network architecture showing inputs: phase-space points p, phase-space mapping variables $y_{ij,k}$ and kinematic invariants s_{ij}. The outputs are the fitted coefficients C_{ijk}.

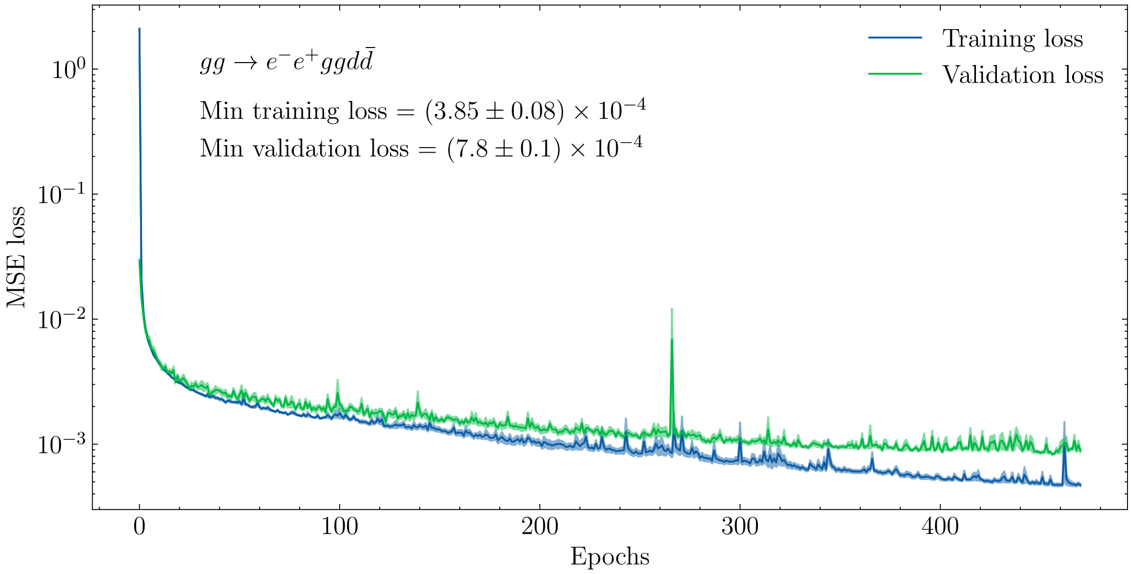


Figure 7.2: Training and validation loss for the models emulating $gg \to e^- e^+ gg d\bar{d}$ matrix elements. We have plotted the mean of the 10 losses as the solid line, with the bands indicating the standard error across these 10 models. The minimum training and validation loss quoted is for the best performing model.

The rationale behind using an ensemble of models is to dampen the effects of random model initialisation, to reduce the effects of stochasticity of the training process, and to provide an estimation of the uncertainty of the neural network prediction. Additionally, the ensembling approach has the advantage of providing, in general, a more robust prediction than a single model, however, that comes at the price of greater evaluation times. In this chapter where we intend to use the trained network in an application where speed is a bottleneck, we have observed that using a single neural network is most performant. This can be attributed to the fact that there is diminishing returns in ensembling models, as there is overlapping information from each replica model. On the other hand, the evaluation time grows linearly with the number of models in the ensemble, meaning a single accurate model strikes a good balance between speed and accuracy. Additionally, the uncertainty associated with the network predictions is not relevant for the application presented in this chapter as the network is used in an intermediary step of a larger calculation.

Since we are interested in reducing evaluation time of the neural network, it is natural to wonder whether it is worth trading off even more accuracy for speed in the form of reducing the size of the neural network. A smaller network with fewer nodes/layers would reduce the size of the weight matrices used during predictions, representing an increase in throughput. However, the neural network is deployed using the highly optimised `ONNX` runtime where we have observed that more compact networks do not reduce evaluation time by very much at all, whereas there is an appreciable decrease in accuracy, leading to lower efficiency of the second unweighting step described below.

7.3 Novel two-stage unweighting algorithm

Generation of unweighted events is carried out via rejection sampling, as outlined in Section 3.2, where event weights are normalised by the maximal event weight, $w_{\max}$, before being compared to a uniform random number $r \in [0, 1]$. If $w/w_{\max} < r$

then events are accepted and assigned a unit-weight. This procedure means that small event weights have a lower probability of being accepted, whereas larger event weights are more likely to be accepted. An issue with unweighting is that the number of accepted events, N, is generally much lower than the number of trial events, N^{trials}. This is encapsulated in the unweighting efficiency

$$\epsilon = \frac{N}{N^{\text{trials}}} \approx \frac{\langle w \rangle_{N^{\text{trials}}}}{w_{\text{max}}} \text{ for large } N^{\text{trials}}, \tag{7.3.1}$$

where $\langle w \rangle_{N^{\text{trials}}}$ is the mean event weight over all trials. The inverse $1/\epsilon$ gives the average number of trial events required to accept one unit-weight event. The rejection sampling algorithm depends on the maximal event weight, w_{max}, which cannot usually be exactly determined given finite statistics. Furthermore, it is possible to have large outlier event weights which when taken to be w_{max} would lead to excessively low unweighting efficiencies.

One method to avoid this issue is to define a reduced w_{max} and accept overweights, meaning weights with $w > w_{\text{max}}$ are accepted but are assigned with a correction factor $\tilde{w} = w/w_{\text{max}}$. This process leads to partially unweighted events and is summarised in Algorithm 2. It is the default method to generate unweighted events in the SHERPA framework. A systematic approach to reducing w_{max} is the per-mille quantile reduction method which defines $w_{\text{max}}^{\text{p.m.}}$ such that overweights contribute at most 0.1% to the total cross-section.

Algorithm 2: Rejection sampling for generating partially unweighted events.

```
while unweighting do
    generate random phase-space point $\mathbf{x}$;
    evaluate event weight $w \leftarrow w(\mathbf{x})$;
    generate uniform random number $r \leftarrow \text{Random}(0, 1)$;
    if $w/w_{\text{max}} > r$ then
        return $\mathbf{x}$ and $\tilde{w} \leftarrow \max(1,\ w/w_{\text{max}})$;
    end
end
```

Whilst the partial unweighting approach increases overall unweighting efficiency, it remains low for high-multiplicity processes. A novel two-step unweighting procedure was introduced in Ref. [103] to accelerate unweighted event generation. This procedure is formed of two separate unweighting steps. The first unweighting step uses a lightweight surrogate neural network model, which approximates the event weight, to rapidly unweight the trial event. Given that the surrogate model is in general imperfect, the second unweighting step corrects for this by calculating the exact event weight and applying a correction factor, meaning the final event accepted is exact and has unit-weight. This second step is similar to the process of dealing with overweights described above. The advantage of this approach is that the expensive exact event weight is only evaluated once the first step accepts a trial event. This means that the number of expensive exact event weight computations is drastically reduced for a sufficiently accurate surrogate model, hence accelerating unweighted event generation. In Ref. [103], the authors used a surrogate model to approximate the full event weight, $w = J_{\mathrm{PS}}|\mathcal{M}|^2$. In this chapter we explore the possibility to emulate only the matrix element using the factorisation-aware model detailed in the previous section, whilst keeping the phase-space weight computation exact.

The two-stage algorithm is outlined in Algorithm 3, where we have denoted the matrix element approximation as $|M|^2$ in line number 3. This two-step approach introduces an additional weight, x_{max}, analogous to the maximal event weight w_{max}. It is the maximum truth-to-prediction ratio. This could in principle be a large value, so the quantile reduction method is used to define a more practical $x_{\mathrm{max}}^{\mathrm{p.m.}}$ for more efficient unweighting in the second step. This will be described in more detail in Section 7.3.1.

The acceleration of this approach can be quantified by the effective gain factor which is given by

$$\begin{aligned} f_{\mathrm{eff}} &= \frac{T_{\mathrm{standard}}}{T_{\mathrm{surrogate}}}\,, \\ &= \frac{N_{\mathrm{full}}^{\mathrm{trials}} \times \langle t_{\mathrm{full}} \rangle}{N_{\mathrm{1st}}^{\mathrm{trials}} \times [\langle t_{\mathrm{surr}} \rangle + \langle t_{\mathrm{PS}} \rangle] + N_{\mathrm{2nd}}^{\mathrm{trials}} \times \langle t_{\mathrm{ME}} \rangle}\,, \\ &= \frac{1}{\frac{\langle t_{\mathrm{surr}} \rangle + \langle t_{\mathrm{PS}} \rangle}{\langle t_{\mathrm{full}} \rangle} \times \frac{\epsilon_{\mathrm{full}}}{\epsilon_{\mathrm{1st}}\epsilon_{\mathrm{2nd}}} + \frac{\langle t_{\mathrm{ME}} \rangle}{\langle t_{\mathrm{full}} \rangle} \times \frac{\epsilon_{\mathrm{full}}}{\epsilon_{\mathrm{2nd}}}}\,. \end{aligned} \tag{7.3.2}$$

Algorithm 3: Two-stage rejection sampling algorithm employing a surrogate model to approximate exact event weight.

```
while unweighting do
    generate phase-space point $\mathbf{x}$;
    calculate approximate event weight $s \leftarrow J_{\mathrm{PS}}|M|^2$;
    generate uniform random number $r_1 \leftarrow \mathrm{Random}(0, 1)$;
    // first unweighting step
    if $s/w_{\mathrm{max}} > r_1$ then
        calculate exact event weight $w \leftarrow w(\mathbf{x})$;
        determine ratio $x \leftarrow w/s$;
        generate uniform random number $r_2 \leftarrow \mathrm{Random}(0, 1)$;
        // second unweighting step
        if $x/x_{\mathrm{max}} > r_2$ then
            return $\mathbf{x}$ and $\widetilde{w} \leftarrow \max(1, s/w_{\mathrm{max}}) \cdot \max(1, x/x_{\mathrm{max}})$
        end
    end
end
```

This effective gain factor can be understood as the ratio of the average times of the standard unweighting approach, compared to the two-stage unweighting approach, such that a higher f_{eff} is desirable. The time spent in the standard approach is simply the number of trial events, $N_{\mathrm{full}}^{\mathrm{trials}}$, multiplied by the average evaluation time of the exact event weight in this sample of trials, $\langle t_{\mathrm{full}} \rangle$. The total time of the two-stage approach can be split up into two steps. The first step involves the evaluation of the surrogate model (including all input computations, pre- and post-processing) and the exact phase-space weight, $\langle t_{\mathrm{surr}} \rangle$ and $\langle t_{\mathrm{PS}} \rangle$, for the number of trial events $N_{\mathrm{1st}}^{\mathrm{trials}}$. The second step involves the evaluation of the exact matrix element (using the phase-space weight from the previous step to recover the exact event weight), $\langle t_{\mathrm{ME}} \rangle$, for $N_{\mathrm{2nd}}^{\mathrm{trials}}$ second step trial events. From the second to the final line of Eq. (7.3.2), we divided the numerator and denominator by $N_{\mathrm{full}}^{\mathrm{trials}} \times \langle t_{\mathrm{full}} \rangle$ and defined

$$\begin{aligned}
\epsilon_{\mathrm{full}} &= \frac{N}{N_{\mathrm{full}}^{\mathrm{trials}}} \approx \frac{\langle w \rangle}{w_{\mathrm{max}}^{\mathrm{p.m.}}} \,, \\
\epsilon_{\mathrm{1st}} &= \frac{N_{\mathrm{2nd}}^{\mathrm{trials}}}{N_{\mathrm{1st}}^{\mathrm{trials}}} \approx \frac{\langle s \rangle}{w_{\mathrm{max}}^{\mathrm{p.m.}}} \,, \\
\epsilon_{\mathrm{2nd}} &= \frac{N}{N_{\mathrm{2nd}}^{\mathrm{trials}}} \approx \frac{\langle x \rangle}{x_{\mathrm{max}}^{\mathrm{p.m.}}} \,,
\end{aligned} \tag{7.3.3}$$

where $\langle w \rangle$ is the mean of the weights generated during SHERPA's integration run (see Section 7.3.1), $\langle s \rangle$ is the mean of the surrogate model predictions on the test set, and $\langle x \rangle$ is the mean of the corresponding truth-to-prediction ratios. These efficiencies represent the unweighting efficiencies of the standard approach, and of the two steps in the two-step approach, respectively. From the expression of the effective gain factor, Eq. (7.3.2), we would expect large gains for processes where ϵ_{full} is low. Indeed, this is the case for the high-multiplicity processes we consider. Furthermore, for a fast and accurate surrogate model, where $\langle t_{\text{surr}} \rangle \ll \langle t_{\text{full}} \rangle$, $\epsilon_{\text{1st}} \approx \epsilon_{\text{full}}$, and $\epsilon_{\text{2nd}} \approx 1$, we would also expect large gains.

7.3.1 Implementation in SHERPA

To implement the neural network emulator described in Section 7.2.3 as a surrogate model for the matrix elements in the two-step algorithm outlined in Section 7.3, we need to generate data for training and testing the neural network, and to use the model to approximate the event weight during the unweighting process. Cross-section predictions in SHERPA are made in two phases. Firstly, there is an optimisation phase which allows the integrator to adapt to the partonic channel of interest [95]. This is followed by an integration phase in which the integrator is used to determine the total cross-section. During this integration phase, weighted events are generated and $w_{\text{max}}^{\text{p.m.}}$ is determined with the quantile reduction method which we describe below. In the standard partial unweighting procedure, the reduced maximal event weight, $w_{\text{max}}^{\text{p.m.}}$, is determined by first sorting weights generated during the integration phase, $\{w_i\}$, such that $w_i < w_{i+1}$, then requiring

$$w_{\text{max}}^{\text{p.m.}} = \min \left(w_j \,\middle|\, \sum_{i=j+1}^{N} w_i < 0.001 \sum_{i=1}^{N} w_i \right) , \tag{7.3.4}$$

i.e. find the first weight from $\{w_i\}$ such that the sum of all larger weights contributes at most 0.1% to the total cross-section.

During the integration phase, 1.5 million events are taken for training and testing data by saving the 4-momenta, phase-space weights, and matrix elements. In this chapter we study the emulation of colour and helicity summed matrix elements, which are evaluated using the AMEGIC matrix element provider in SHERPA. From these 1.5 million events, we take 800k events for training, 200k events for validation during training, and use the final 500k events for independent testing of the model once it has been trained. The testing dataset is used to determine the value of $x_{\max}$. The same quantile reduction method described above can be used to define a reduced $x_{\max}$, $x_{\max}^{\text{p.m}}$, that allows for more efficient unweighting in the second step of the two-step procedure at the expense of overweights. By sorting the sequence of truth-to-prediction ratios, $\{x_i\}$, such that $x_i < x_{i+1}$ and using the same sorting order for $\{s_i\}$, $x_{\max}^{\text{p.m.}}$ is defined as

$$x_{\max}^{\text{p.m.}} = \min \left(x_j \, \middle| \, \sum_{i=j+1}^{N} x_i s_i < 0.001 \sum_{i=1}^{N} x_i s_i \right) . \tag{7.3.5}$$

This process is repeated for each partonic channel that we study, where each channel has its own neural network surrogate model. The `ONNX` runtime has a simple `C++` API, meaning it is straightforward to incorporate the model into a `C++` workflow. The interface to the SHERPA framework is then a case of writing wrapper code that takes the phase-space point generated from SHERPA to compute the model input variables y_{ijk} and s_{ij}, make the model predictions, C_{ijk}, and then combines them with the appropriate dipole functions to form an approximation of the matrix element. This setup can then be used to perform the two-stage unweighting procedure in Algorithm 3 by calling the surrogate model for every $|M|^2$ evaluation.

7.4 Results

In this section, we evaluate the performance of the neural network model by examining the error distributions for the predictions made with the model. Following this, we display results for the application in the two-stage unweighting procedure by quoting effective gain factors, f_{eff}.

7.4.1 Emulator error distributions

In Figure 7.3, we show error distributions for the neural network predictions, by plotting the prediction-to-truth ratio, w/s, for the 500k events in the independent testing dataset for each channel, as indicated by the legend labels. In the left-hand subplots, we plot w/s on a linear scale to assess the model performance in terms of accuracy, whereas on the right-hand subplots we plot $\log_{10}(w/s)$. This allows us to examine the tails of the distribution which are important for the second unweighting step, as a large right-handed tail skews the value of $x^{\text{p.m.}}_{\text{max}}$. All error distributions are centred around the ideal values of 1 or 0, for the left- and right-hand subplots, respectively, with the distributions having a Gaussian-like distribution that is narrow.

We see that performance on the channels associated with Z+jets production are generally worse than for the top quark pair production channels. This can be attributed to the multiplicity of the channels, as increasing multiplicity generally leads to a drop in accuracy as already discussed in Chapters 5 and 6. At the same multiplicity, the $gg \to t\bar{t}ggg$ channel presents a more difficult channel to model than $u\bar{u} \to t\bar{t}gd\bar{d}$ due to the presence of more infrared singularities, however, the associated accuracy with this gluon initiated channel remains higher than for the $t\bar{t} + 4j$ case, in line with the assessment above.

To inspect the tails on the right-hand subplots further, we plot 2d histograms of w/s against the value of the event weights. This allows us to see whether the tails emerge from large weight events, or from small weight events. This is illustrated in Figure 7.4 for the $Z + 5j$ process as this is the channel with the lowest accuracy, meaning the other channels are better behaved. We see that the model is generally well-behaved for large event weights as the high population bins depicted in yellow, are centred around the ideal value of 0, and the error distribution is generally contained inside a band. From this plot, we also see that outlier events depicted in purple are predominately smaller event weights.

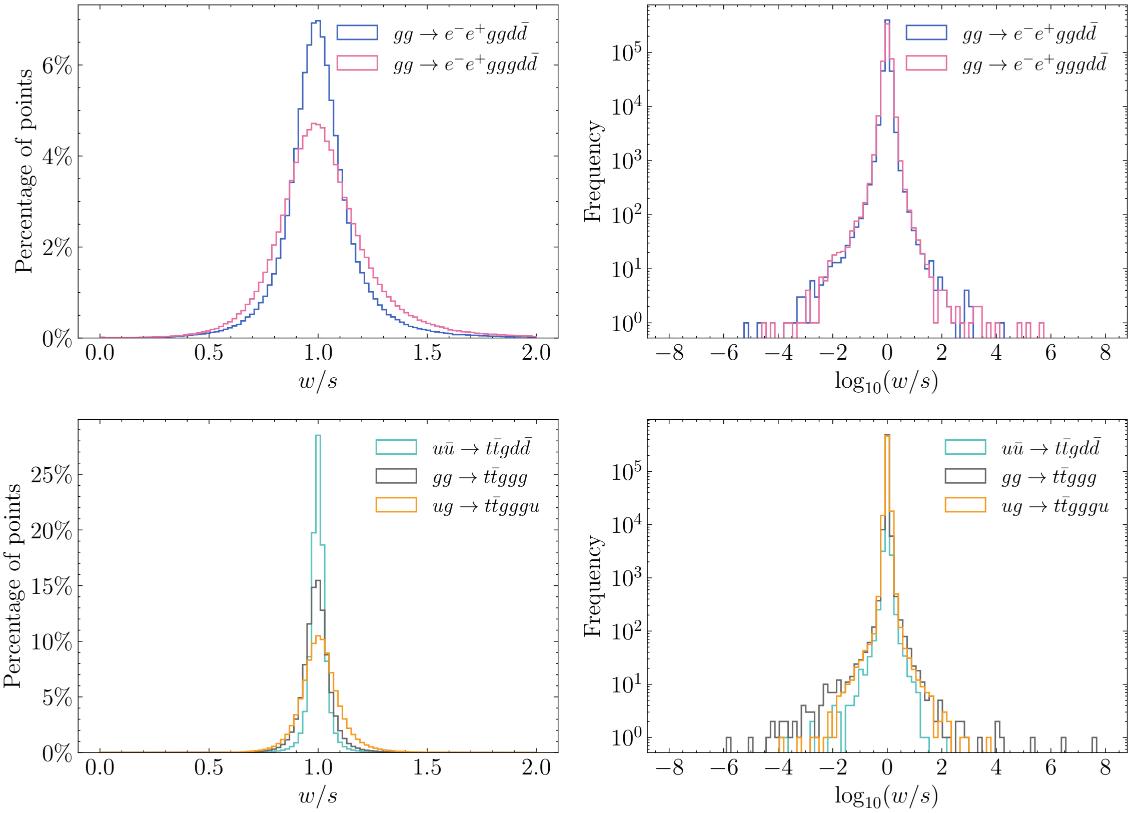


Figure 7.3: Error distributions for all partonic channels. All histograms are produced from 500k test events. Left: linear truth-to-prediction ratio distributions, right: $\log_{10}$ truth-to-prediction ratio distributions. The upper two subplots illustrate the $Z + 4j$ and $Z + 5j$ processes, with the lower two subplots illustrating the $t\bar{t} + 3j$ and $t\bar{t} + 4j$ processes.

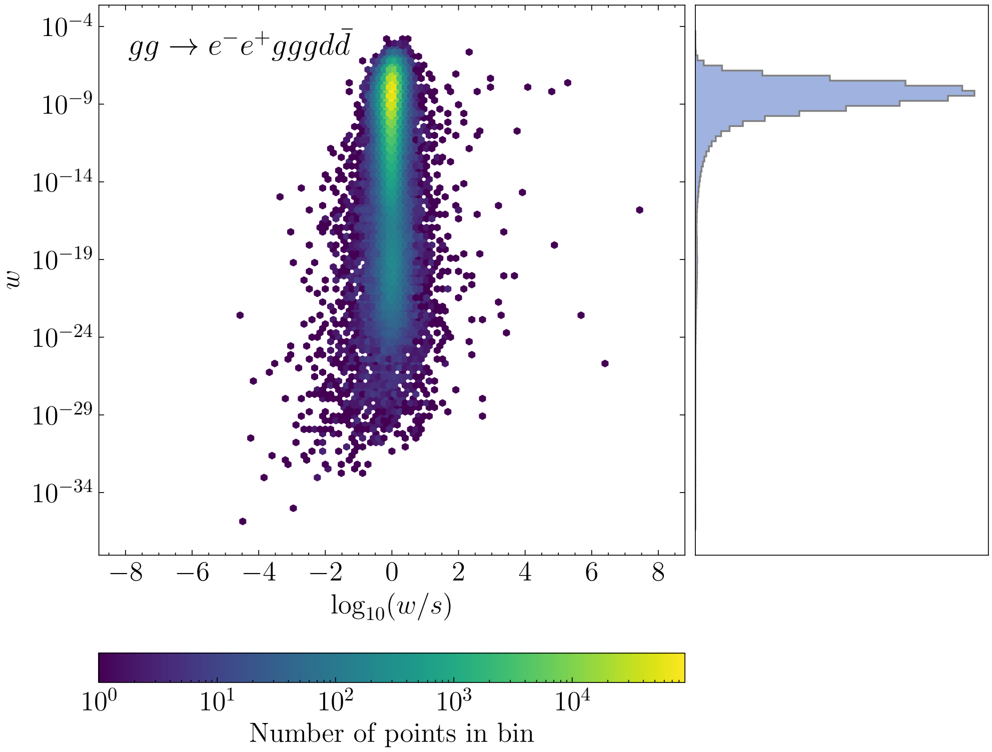


Figure 7.4: Left: 2d histogram of $\log_{10}(w/s)$ against the respective event weight, w for the $gg \to e^-e^+gggd\bar{d}$ channel. Yellow bins represent high population bins, and purple bins represent single points. Right: marginal distribution of $gg \to e^-e^+gggd\bar{d}$ event weights.

Process	Timings / s			Efficiencies			f_{eff}
	$\langle t_{\text{ME}}\rangle$	$\langle t_{\text{PS}}\rangle$	$\langle t_{\text{surr}}\rangle$	ϵ_{full}	ϵ_{1st}	ϵ_{2nd}	
$gg \to e^- e^+ ggd\bar{d}$	5.4×10^{-2}	4.0×10^{-4}	1.4×10^{-4}	1.6×10^{-2}	1.4×10^{-2}	0.37	14
$gg \to e^- e^+ gggd\bar{d}$	1.6×10^{1}	5.7×10^{-3}	2.0×10^{-4}	7.6×10^{-4}	8.5×10^{-4}	0.29	269
$u\bar{u} \to t\bar{t}gd\bar{d}$	5.3×10^{-3}	3.5×10^{-5}	1.4×10^{-4}	8.7×10^{-4}	2.5×10^{-3}	0.28	23
$gg \to t\bar{t}ggg$	3.3	9.0×10^{-4}	1.8×10^{-4}	8.8×10^{-3}	1.5×10^{-2}	0.50	55
$ug \to t\bar{t}gggu$	5.1×10^{1}	4.0×10^{-3}	2.4×10^{-4}	1.5×10^{-3}	1.6×10^{-3}	0.57	366

Table 7.4: Unweighting performance measures for all partonic channels.

7.4.2 Unweighting gains

With the model performance validated, we move on to discussing the acceleration of unweighted event generation by using the neural network emulator as the surrogate model in the two-stage unweighting procedure.

The amount of acceleration is quantified by the effective gain factor, f_{eff}, which measures the average time saved to unweight a sample of events compared to the standard one-step rejection sampling approach. To determine f_{eff}, we run through the procedure described in Algorithm 3 for a number of events in the testing datasets and record $\langle t_{\text{surr}}\rangle$, $\langle t_{\text{PS}}\rangle$, and $\langle t_{\text{ME}}\rangle$. We note that $\langle t_{\text{full}}\rangle = \langle t_{\text{PS}}\rangle + \langle t_{\text{ME}}\rangle$. Depending on the complexity of the process, between 10 and 10000 events are unweighted to get a reliable estimate of these times. Eqs. (7.3.2) and (7.3.3) are then used to calculate f_{eff}.

In Table 7.4 we quote the timings, efficiencies, and the computed effective gain factors for all channels, where we can see the speed up ranges from 14 to up to 366 times compared to the standard approach. This is due to the rapid neural network evaluation times and good predictive accuracy leading to relatively high ϵ_{1st} and ϵ_{2nd}. Interestingly, the largest gains are seen for the most computationally expensive processes, namely, the $Z + 5j$ and $t\bar{t} + 4j$ processes. This demonstrates that the surrogate model has managed to accurately model the matrix elements, and to produce the approximations rapidly. We can see that $\langle t_{\text{surr}}\rangle$ only scales weakly with the multiplicity, whereas $\langle t_{\text{ME}}\rangle$ increases very rapidly for more complex final-states.

7.5 Conclusion

In this chapter we have studied the extension of the factorisation-aware model from electron-positron annihilation matrix elements explored in Chapters 5 and 6 to hadronic collisions. The model was readily adapted by changing the ingredients in the ansatz of the matrix element to include the full set of massless dipoles to account for radiation from the initial-state partons. Additionally, we included the full set of massive dipole functions to enable the accurate modelling of matrix elements with massive partons, in this case top quark pair production. We have demonstrated that with these additional dipole functions, it is possible to accurately approximate the matrix elements of high-multiplicity hadronic processes with well-behaved predictions for large event weights.

We have also showcased an application of the neural network model in the form of accelerating event unweighting in the SHERPA framework by using the neural network model in a novel two-stage unweighting procedure. The performance of this unweighting procedure relied on the fast and accurate predictions of the model. We showed that for all partonic channels considered, the unweighting procedure was accelerated, with the largest gains coming from the most computationally expensive channels where the effective gain factors were well over two orders of magnitude.

Chapter 8

Conclusion and outlook

In this thesis we have presented a physics-inspired neural network model to emulate matrix elements. The application of using a neural network to fit functions is a common scenario, however, the unique singularity structure of physical matrix elements precludes a naive approach if accuracy is desired. Indeed, the per-point accuracy of matrix elements is generally desired, especially in the infrared regions of phase-space where the emergence of large weights can contribute significantly to the total cross-section. Fortunately, matrix elements factorise in these infrared regions of phase-space, with well-understood explicit functional forms describing the single soft and collinear limits. By incorporating these functions into the neural network model via an ansatz composed of a linear combination of these singular functions, it is possible to transform the problem of fitting a potentially rapidly changing function over phase-space to the fitting of more well-behaved coefficients. We dubbed this family of models factorisation-aware as the neural network learns to choose the relevant functional behaviours in specific soft and collinear limits where the fitted coefficients are constrained. Outside of these limits, we make use of the well documented fitting capabilities of neural networks to form an interpolation.

We first demonstrated the use of the factorisation-aware model in Chapter 5 where we utilised the Catani-Seymour dipole functions as the singular functions in the ansatz to model tree-level electron-positron annihilation matrix elements. It was shown that by building these functions into the emulator, there was drastic improvements in per-point accuracy compared to existing methods, with accuracy well below 1% for all processes studied. Furthermore, the inclusion of the singular functions describing infrared behaviour of the matrix elements allowed well-behaved extrapolation of the model into untrained regions of phase-space which were potentially more singular than those seen previously during training.

The success of this approach prompted the study of applying the factorisation-aware model to emulate the more expensive one-loop matrix elements for the same family of processes, as seen in Chapter 6. It was found that by adapting the Catani-Seymour dipole functions to the more relevant antenna functions containing one-loop singular behaviour, the accurate emulation of NLO QCD k-factors was achieved. In this study we also demonstrated that it was possible for the emulator to learn the renormalisation scale dependence of the one-loop matrix elements, allowing it to be used to carry out scale variations. We showed that $e^+e^- \to q\bar{q}ggg$ NLO k-factors could be predicted with accuracy at the 1% level, representing a speed up in evaluation of over 3 orders of magnitude compared to MadGraph on a single CPU core.

The study of electron-positron annihilation matrix elements whilst important for planned future lepton-lepton colliders, is not critical for making predictions for the LHC. For that, hadron-hadron initiated matrix elements are much more relevant. In Chapter 7, we extended the factorisation-aware model to hadronic collisions by incorporating the full set of massless and massive dipole functions. The inclusion of these functions into the emulation model allowed us to model $Z + \{4, 5\}$ jets and $t\bar{t} + \{3, 4\}$ jets matrix elements at leading order, which represent computationally expensive and phenomenologically relevant processes. We found that the accuracy of the emulator, while lower than in the cleaner e^+e^- environment, was not surprising given that accuracy scales inversely with multiplicity and these were the highest multiplicity processes considered in this thesis. However, the well-behaved predictions in the infrared limits and the low population tails makes it suitable for use as a surrogate model for event weights in the context of event unweighting, where stability of predictions is desirable.

We showed that using the emulator in a novel two-stage unweighting procedure, where small mismatch from the full event weight is corrected for, could reduce the time spent in unweighting a sample of events. We saw that the unweighting of events could be accelerated by well over 2 orders of magnitude for the most computationally expensive processes, $Z + 5$ jets and $t\bar{t} + 4$ jets.

The factorisation-aware model presented in this thesis has been demonstrated to be a powerful tool to emulate matrix elements for a variety of processes, at tree-level and one-loop level, owing to the flexibility of the modelling philosophy. The path forward is clear: to reliably accelerate the generation of simulated unit-weight events required for LHC experiments, the modelling procedure has to be extended to arbitrary SM processes, both for colour-summed, and colour-sampled matrix elements, with an easy-to-use interface to general purpose Monte Carlo event generators. We have taken the first steps towards this by proving the viability of the model in a production setting by deploying it within the Sherpa framework to successfully accelerate the process of unweighting events.

Appendix A

Catani-Seymour notation

This appendix serves as supplementary material for Chapter 2, specifically, for the tree-level matrix element factorisation in Section 2.2 and Catani-Seymour dipoles in Section 2.4.

A.1 Notation

The notation for matrix elements given in Ref. [51] is repeated here for convenience.

Consider a tree-level matrix element with n partons in the final-state

$$\mathcal{M}_n^{c_1,\ldots,c_n;s_1,\ldots,s_n}(p_1,\ldots,p_n)\,, \tag{A.1.1}$$

where $\{c_1,\ldots,c_n\}$ are colour indices, $\{s_1,\ldots,s_n\}$ are spin indices, and $\{p_1,\ldots,p_n\}$ are momenta.

Introducing the basis $\{|c_1,\ldots,c_n\rangle \otimes |s_1,\ldots,s_n\rangle\}$ in colour + helicity space allows us to write

$$\mathcal{M}_n^{c_1,\ldots,c_n;s_1,\ldots,s_n}(p_1,\ldots,p_n) \equiv \left(\langle c_1,\ldots,c_n| \otimes \langle s_1,\ldots,s_n|\right)|1,\ldots,n\rangle_n\,, \tag{A.1.2}$$

where $|1,\ldots,n\rangle_n$ is a vector in colour + helicity space. The indices inside the vector denote the spin and colour indices, while the subscript denotes the multiplicity of the matrix element. The matrix element squared, summed over final-state colours and spins, can therefore be written as

$$|\mathcal{M}_n|^2 = {}_n\langle 1, \ldots, n | 1, \ldots, n \rangle_n \,. \tag{A.1.3}$$

In the case of partons in the initial-state the colour + helicity vector $|1, \ldots, n\rangle$ has to be normalised by a factor $\sqrt{n_c}$ for each initial state parton. For hadron-hadron collisions this amounts to

$$|1, \ldots, n\rangle_n \to \frac{1}{\sqrt{n_c(a) n_c(b)}} |1, \ldots, n\rangle_n \,, \tag{A.1.4}$$

where $n_c(a)$ denotes the number of colour states of parton a. Namely $n_c(q) = n_c(\bar{q}) = N_c$ and $n_c(g) = N_c^2 - 1$.

The colour-charge operators $\boldsymbol{T}_i$ act on colour space to give the colour-correlated matrix element

$$\begin{aligned} {}_n\langle 1, \ldots, n; a, b | \boldsymbol{T}_I \cdot \boldsymbol{T}_J | 1, \ldots, n; a, b\rangle_n = \\ \frac{1}{n_c(a) n_c(b)} \left[\mathcal{M}_n^{c_1, \ldots, c_I, \ldots, c_J, \ldots, c_n}(p_1, \ldots, p_n; p_a, p_b)\right]^* \\ \times \mathcal{T}^e_{c_I d_I} \mathcal{T}^e_{c_J d_J} \times \left[\mathcal{M}^{d_1, \ldots, d_I, \ldots, d_J, \ldots, d_n}(p_1, \ldots, p_n; p_a, p_b)\right] \,, \end{aligned} \tag{A.1.5}$$

where I and J denotes initial- or final-state partons. For a parton I, the matrices $\mathcal{T}^e_{c_I, d_I}$ are defined as

$$\mathcal{T}^e_{c_I d_I} = \begin{cases} -i f_{cde} & \text{if} \quad I = \text{gluon}\,, \\ \;\; T^e_{cd} & \text{if} \quad I = \text{final-state quark or initial-state antiquark}\,, \\ -T^e_{cd} & \text{if} \quad I = \text{final-state antiquark or initial-state quark}\,. \end{cases} \tag{A.1.6}$$

Each colour vector $|1, \ldots, n; a, b\rangle$ is a colourless state, meaning colour conservation can be written as

$$\sum_I \boldsymbol{T}_I |1, \ldots, n; a, b\rangle = 0 \,. \tag{A.1.7}$$

Additionally, the colour-charge operators have the following properties

$$\boldsymbol{T}_I \cdot \boldsymbol{T}_J = \boldsymbol{T}_J \cdot \boldsymbol{T}_I \quad \text{for} \quad I \neq J \,,$$

$$\begin{aligned} \boldsymbol{T}_q^2 &= \boldsymbol{T}_{\bar{q}}^2 = C_F \,, \\ \boldsymbol{T}_g^2 &= C_A \,. \end{aligned} \tag{A.1.8}$$

Appendix B

Appendices for Chapter 5

B.1 Phase-space trajectories

We can see the RAMBO algorithm as a map from the unit hypercube in some high dimension into the n-particle phase-space. A flat distribution in the hypercube maps to a flat distribution in the multi-particle phase-space. To generate our phase-space trajectories we pick two points at random in the unit hypercube and map the line between them using the RAMBO mapping. As a result, each point on the resulting trajectory has equal probability density in phase-space. Any other phase-space generator that smoothly maps the unit hypercube to a multi-particle phase-space could replace RAMBO in this procedure and could lead to trajectories with very different characteristics. One could for example imagine a sophisticated algorithm that only maps points close to the boundary of the hypercube to soft or collinear configurations. Using such an algorithm would have trajectories that avoid configurations with many particles more collinear or soft than the end-points of the trajectories. We find that with RAMBO the trajectories tend not to avoid difficult phase-space configurations and are therefore a good test of the extrapolation properties of our method. Figure B.1 shows the trajectory we chose in Section 5.3.3.

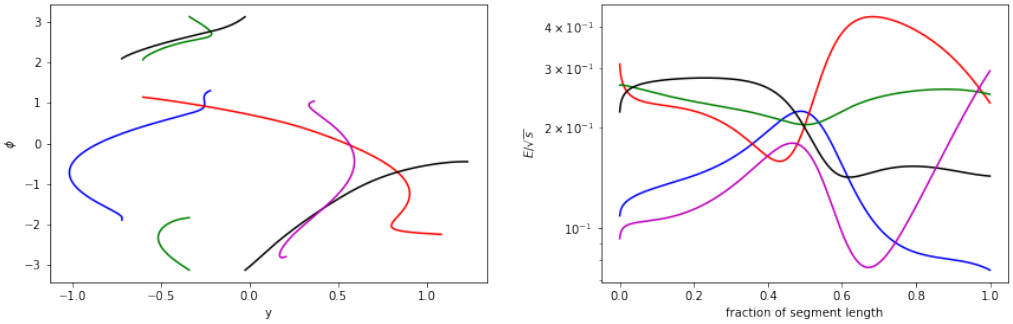


Figure B.1: Left: rapidity and azimuthal angle trajectories for the five final state particles in the trajectories used in Section 5.3.3. Right: Evolution of the same particle energies as a function of the position along the segment between the two random points in the unit hypercube.

B.2 Azimuthal angle ϕ_{ij} calculation

To calculate the azimuthal angle ϕ_{ij} for a pair of particle momenta p_i and p_j we first consider the plane perpendicular to the momentum

$$\vec{p}_{ij} = \vec{p}_i + \vec{p}_j \, . \tag{B.2.1}$$

We project the unit vector in the z direction and the momentum of particle i onto this plane[20]:

$$\vec{r}_z = \vec{e}_z - \left(\frac{\vec{p}_{ij} \cdot \vec{e}_z}{\vec{p}_{ij}^{\,2}} \right) \vec{p}_{ij} \, , \tag{B.2.2}$$

$$\vec{r}_i = \vec{p}_i - \left(\frac{\vec{p}_{ij} \cdot \vec{p}_i}{\vec{p}_{ij}^{\,2}} \right) \vec{p}_{ij} \, . \tag{B.2.3}$$

The angle ϕ_{ij} is the angle between these two projected vectors.

$$\sin \phi_{ij} = \hat{r}_{ij} \cdot (\hat{r}_i \times \hat{r}_z) \; , \quad \cos \phi_{ij} = \hat{r}_i \cdot \hat{r}_z \, , \tag{B.2.4}$$

where we have normalised all vectors to be unit vectors:

$$\hat{r}_z = \frac{\vec{r}_z}{|r_z|} \; , \qquad \hat{r}_i = \frac{\vec{r}_i}{|r_i|} \; , \qquad \hat{r}_{ij} = \frac{\vec{p}_{ij}}{|p_{ij}|} \; . \tag{B.2.5}$$

[20] using particle j instead results in a shift of ϕ_{ij} by π which makes no difference for $\sin 2\phi$ or $\cos 2\phi$.

B.3 Jensen's Inequality

Jensen's inequality states that for concave functions

$$f(\mathbb{E}[Y]) \geq \mathbb{E}[f(Y)]\,, \tag{B.3.1}$$

where $\mathbb{E}$ is the expectation value, the function f in our case is arcsinh that behaves similarly to the natural logarithm due to the scale of the problem, and Y is a random variable representing our target distribution. This inequality can be rewritten as

$$f(\mathbb{E}[Y]) - \mathbb{E}[f(Y)] \geq 0\,, \tag{B.3.2}$$

which is known as Jensen's gap. With a concave function, the mean of the transformed target distribution will always be underestimating the actual mean, i.e.

$$\mathbb{E}[Y] \geq \sinh\left(\mathbb{E}[\text{arcsinh}(y)]\right)\,, \tag{B.3.3}$$

where the LHS is the actual expectation value (cross-section) of the random variable Y and the RHS is the shifted expectation value that the neural network learns. In our scenario the neural network reduces the variance of the residual distribution, meaning the gap in reality is small but there will always be an offset in the mean value learnt.

B.4 Additional random phase-space trajectory

Here we present another random phase-space trajectory similar to the one in Figure 5.7 to demonstrate that features of that figure are not unique.

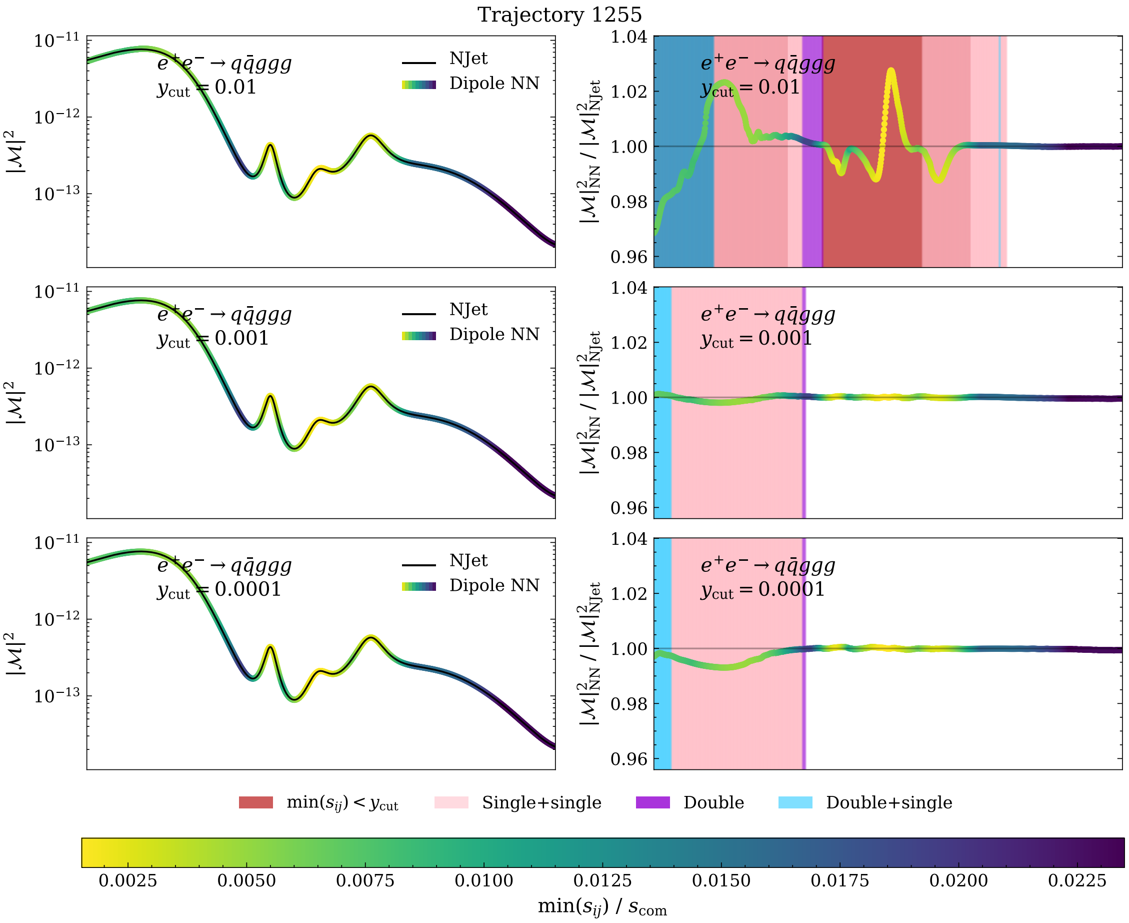


Figure B.2: Another random phase-space trajectory where the 'double+single' regions of phase-space are explicitly shown in blue.

Bibliography

[1] "File:Standard Model of Elementary Particles.svg - Wikimedia Commons." https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg.

[2] Particle Data Group collaboration, R. L. Workman, *Review of Particle Physics*, *PTEP* **2022** (2022) 083C01.

[3] A. LeNail, *Nn-svg: Publication-ready neural network architecture schematics*, *Journal of Open Source Software* **4** (2019) 747.

[4] S. Badger and J. Bullock, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes*, *JHEP* **06** (2020) 114, [2002.07516].

[5] D. Maître and H. Truong, *A factorisation-aware Matrix element emulator*, *JHEP* **11** (2021) 066, [2107.06625].

[6] J. Aylett-Bullock, C. Cuesta-Lazaro, A. Quera-Bofarull, M. Icaza-Lizaola, A. Sedgewick, H. Truong et al., *June: open-source individual-based epidemiology simulation*, *Royal Society Open Science* **8** (2021) 210506.

[7] I. Vernon, J. Owen, J. Aylett-Bullock, C. Cuesta-Lazaro, J. Frawley, A. Quera-Bofarull et al., *Bayesian emulation and history matching of june*, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **380** (2022) 20220039.

[8] T. Aoyama, T. Kinoshita and M. Nio, *Revised and Improved Value of the QED Tenth-Order Electron Anomalous Magnetic Moment*, *Phys. Rev. D* **97** (2018) 036001, [1712.06060].

[9] ATLAS collaboration, G. Aad et al., *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, *Phys. Lett. B* **716** (2012) 1–29, [1207.7214].

[10] CMS collaboration, S. Chatrchyan et al., *Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC*, *Phys. Lett. B* **716** (2012) 30–61, [1207.7235].

[11] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, *Phys. Rev. Lett.* **13** (1964) 321–323.

[12] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, *Phys. Rev. Lett.* **13** (1964) 508–509.

[13] G. S. Guralnik, C. R. Hagen and T. W. B. Kibble, *Global Conservation Laws and Massless Particles*, *Phys. Rev. Lett.* **13** (1964) 585–587.

[14] Super-Kamiokande collaboration, Y. Fukuda et al., *Evidence for oscillation of atmospheric neutrinos*, *Phys. Rev. Lett.* **81** (1998) 1562–1567, [hep-ex/9807003].

[15] SNO collaboration, Q. R. Ahmad et al., *Direct evidence for neutrino flavor transformation from neutral current interactions in the Sudbury Neutrino Observatory*, *Phys. Rev. Lett.* **89** (2002) 011301, [nucl-ex/0204008].

[16] M. E. Peskin and D. V. Schroeder, *An Introduction to quantum field theory*. Addison-Wesley, Reading, USA, 1995.

[17] M. D. Schwartz, *Quantum Field Theory and the Standard Model*. Cambridge University Press, 3, 2014.

[18] J. C. Romao and J. P. Silva, *A resource for signs and Feynman diagrams of the Standard Model*, *Int. J. Mod. Phys. A* **27** (2012) 1230025, [1209.6213].

[19] M. Gell-Mann, *Symmetries of baryons and mesons*, *Phys. Rev.* **125** (1962) 1067–1084.

[20] J. Collins, *Foundations of perturbative QCD*, vol. 32. Cambridge University Press, 11, 2013.

[21] J. C. Collins and D. E. Soper, *The Theorems of Perturbative QCD*, *Ann. Rev. Nucl. Part. Sci.* **37** (1987) 383–409.

[22] J. C. Collins, D. E. Soper and G. F. Sterman, *Factorization of Hard Processes in QCD*, *Adv. Ser. Direct. High Energy Phys.* **5** (1989) 1–91, [hep-ph/0409313].

[23] S. Amoroso et al., *Snowmass 2021 whitepaper: Proton structure at the precision frontier*, 2203.13923.

[24] J. J. Ethier and E. R. Nocera, *Parton Distributions in Nucleons and Nuclei*, *Ann. Rev. Nucl. Part. Sci.* **70** (2020) 43–76, [2001.07722].

[25] P. Jimenez-Delgado, W. Melnitchouk and J. F. Owens, *Parton momentum and helicity distributions in the nucleon*, *J. Phys. G* **40** (2013) 093102, [1306.6515].

[26] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht et al., *LHAPDF6: parton density access in the LHC precision era*, *Eur. Phys. J. C* **75** (2015) 132, [1412.7420].

[27] PDF4LHC Working Group collaboration, R. D. Ball et al., *The PDF4LHC21 combination of global PDF fits for the LHC Run III*, *J. Phys. G* **49** (2022) 080501, [2203.05506].

[28] T.-J. Hou et al., *New CTEQ global analysis of quantum chromodynamics with high-precision data from the LHC*, *Phys. Rev. D* **103** (2021) 014013, [1912.10053].

[29] S. Bailey, T. Cridge, L. A. Harland-Lang, A. D. Martin and R. S. Thorne, *Parton distributions from LHC, HERA, Tevatron and fixed target data: MSHT20 PDFs*, *Eur. Phys. J. C* **81** (2021) 341, [2012.04684].

[30] NNPDF collaboration, R. D. Ball et al., *The path to proton structure at 1% accuracy*, *Eur. Phys. J. C* **82** (2022) 428, [2109.02653].

[31] Y. L. Dokshitzer, *Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics.*, *Sov. Phys. JETP* **46** (1977) 641–653.

[32] V. N. Gribov and L. N. Lipatov, *Deep inelastic e p scattering in perturbation theory*, *Sov. J. Nucl. Phys.* **15** (1972) 438–450.

[33] L. N. Lipatov, *The parton model and perturbation theory*, *Yad. Fiz.* **20** (1974) 181–198.

[34] G. Altarelli and G. Parisi, *Asymptotic Freedom in Parton Language*, *Nucl. Phys. B* **126** (1977) 298–318.

[35] S. Moch, J. A. M. Vermaseren and A. Vogt, *The Three loop splitting functions in QCD: The Nonsinglet case*, *Nucl. Phys. B* **688** (2004) 101–134, [hep-ph/0403192].

[36] A. Vogt, S. Moch and J. A. M. Vermaseren, *The Three-loop splitting functions in QCD: The Singlet case*, *Nucl. Phys. B* **691** (2004) 129–181, [hep-ph/0404111].

[37] E. A. Kuraev and V. S. Fadin, *On Radiative Corrections to e+ e- Single Photon Annihilation at High-Energy*, *Sov. J. Nucl. Phys.* **41** (1985) 466–472.

[38] G. 't Hooft and M. J. G. Veltman, *Regularization and Renormalization of Gauge Fields*, *Nucl. Phys. B* **44** (1972) 189–213.

[39] M. L. Mangano, *Introduction to qcd*, 1999.

[40] C. G. Callan, Jr., *Broken scale invariance in scalar field theory*, *Phys. Rev. D* **2** (1970) 1541–1547.

[41] K. Symanzik, *Small distance behavior in field theory and power counting*, *Commun. Math. Phys.* **18** (1970) 227–246.

[42] P. A. Baikov, K. G. Chetyrkin and J. H. Kühn, *Five-Loop Running of the QCD coupling constant*, *Phys. Rev. Lett.* **118** (2017) 082002, [1606.08659].

[43] T. Luthe, A. Maier, P. Marquard and Y. Schroder, *The five-loop Beta function for a general gauge group and anomalous dimensions beyond Feynman gauge*, *JHEP* **10** (2017) 166, [1709.07718].

[44] D. J. Gross and F. Wilczek, *Ultraviolet Behavior of Nonabelian Gauge Theories*, *Phys. Rev. Lett.* **30** (1973) 1343–1346.

[45] H. D. Politzer, *Reliable Perturbative Results for Strong Interactions?*, *Phys. Rev. Lett.* **30** (1973) 1346–1349.

[46] F. Bloch and A. Nordsieck, *Note on the Radiation Field of the electron*, *Phys. Rev.* **52** (1937) 54–59.

[47] D. R. Yennie, S. C. Frautschi and H. Suura, *The infrared divergence phenomena and high-energy processes*, *Annals Phys.* **13** (1961) 379–452.

[48] T. Kinoshita, *Mass singularities of Feynman amplitudes*, *J. Math. Phys.* **3** (1962) 650–677.

[49] T. D. Lee and M. Nauenberg, *Degenerate Systems and Mass Singularities*, *Phys. Rev.* **133** (1964) B1549–B1562.

[50] G. P. Salam, *Towards Jetography*, *Eur. Phys. J. C* **67** (2010) 637–686, [0906.1833].

[51] S. Catani and M. H. Seymour, *A General algorithm for calculating jet cross-sections in NLO QCD*, *Nucl. Phys. B* **485** (1997) 291–419, [hep-ph/9605323].

[52] A. Gehrmann-De Ridder, T. Gehrmann and E. W. N. Glover, *Antenna subtraction at NNLO*, *JHEP* **09** (2005) 056, [hep-ph/0505111].

[53] D. A. Kosower, *Antenna factorization of gauge theory amplitudes*, *Phys. Rev. D* **57** (1998) 5410–5416, [hep-ph/9710213].

[54] J. M. Campbell, M. A. Cullen and E. W. N. Glover, *Four jet event shapes in electron - positron annihilation*, *Eur. Phys. J. C* **9** (1999) 245–265, [hep-ph/9809429].

[55] Z. Bern, L. J. Dixon, D. C. Dunbar and D. A. Kosower, *One loop n point gauge theory amplitudes, unitarity and collinear limits*, *Nucl. Phys. B* **425** (1994) 217–260, [hep-ph/9403226].

[56] Z. Bern and G. Chalmers, *Factorization in one loop gauge theory*, *Nucl. Phys. B* **447** (1995) 465–518, [hep-ph/9503236].

[57] Z. Bern, V. Del Duca and C. R. Schmidt, *The Infrared behavior of one loop gluon amplitudes at next-to-next-to-leading order*, *Phys. Lett. B* **445** (1998) 168–177, [hep-ph/9810409].

[58] D. A. Kosower, *All order collinear behavior in gauge theories*, *Nucl. Phys. B* **552** (1999) 319–336, [hep-ph/9901201].

[59] Z. Bern, V. Del Duca, W. B. Kilgore and C. R. Schmidt, *The infrared behavior of one loop QCD amplitudes at next-to-next-to leading order*, *Phys. Rev. D* **60** (1999) 116001, [hep-ph/9903516].

[60] K. Fabricius, I. Schmitt, G. Kramer and G. Schierholz, *Higher Order Perturbative QCD Calculation of Jet Cross-Sections in e+ e- Annihilation*, *Z. Phys. C* **11** (1981) 315.

[61] G. Kramer and B. Lampe, *Jet Cross-Sections in e+ e- Annihilation*, *Fortsch. Phys.* **37** (1989) 161.

[62] W. T. Giele and E. W. N. Glover, *Higher order corrections to jet cross-sections in e+ e- annihilation*, *Phys. Rev. D* **46** (1992) 1980–2010.

[63] T. Binoth and G. Heinrich, *An automatized algorithm to compute infrared divergent multiloop integrals*, *Nucl. Phys. B* **585** (2000) 741–759, [hep-ph/0004013].

[64] T. Binoth and G. Heinrich, *Numerical evaluation of multiloop integrals by sector decomposition*, *Nucl. Phys. B* **680** (2004) 375–388, [hep-ph/0305234].

[65] R. K. Ellis, D. A. Ross and A. E. Terrano, *Calculation of Event Shape Parameters in e+ e- Annihilation*, *Phys. Rev. Lett.* **45** (1980) 1226–1229.

[66] D. A. Kosower, *Multiple singular emission in gauge theories*, *Phys. Rev. D* **67** (2003) 116003, [hep-ph/0212097].

[67] S. Catani, S. Dittmaier, M. H. Seymour and Z. Trocsanyi, *The Dipole formalism for next-to-leading order QCD calculations with massive partons*, *Nucl. Phys. B* **627** (2002) 189–265, [hep-ph/0201036].

[68] S. Frixione, Z. Kunszt and A. Signer, *Three jet cross-sections to next-to-leading order*, *Nucl. Phys. B* **467** (1996) 399–442, [hep-ph/9512328].

[69] S. Frixione, *A General approach to jet cross-sections in QCD*, *Nucl. Phys. B* **507** (1997) 295–314, [hep-ph/9706545].

[70] D. A. Kosower, *Antenna factorization in strongly ordered limits*, *Phys. Rev. D* **71** (2005) 045016, [hep-ph/0311272].

[71] W. J. Torres Bobadilla et al., *May the four be with you: Novel IR-subtraction methods to tackle NNLO calculations*, *Eur. Phys. J. C* **81** (2021) 250, [2012.02567].

[72] S. Catani, S. Dittmaier and Z. Trocsanyi, *One loop singular behavior of QCD and SUSY QCD amplitudes with massive partons*, *Phys. Lett. B* **500** (2001) 149–160, [hep-ph/0011222].

[73] A. Gehrmann-De Ridder, T. Gehrmann and E. W. N. Glover, *Infrared structure of e+ e- —> 2 jets at NNLO*, *Nucl. Phys. B* **691** (2004) 195–222, [hep-ph/0403057].

[74] A. Daleo, T. Gehrmann and D. Maitre, *Antenna subtraction with hadronic initial states*, *JHEP* **04** (2007) 016, [hep-ph/0612257].

[75] A. Daleo, A. Gehrmann-De Ridder, T. Gehrmann and G. Luisoni, *Antenna subtraction at NNLO with hadronic initial states: initial-final configurations*, *JHEP* **01** (2010) 118, [0912.0374].

[76] R. Boughezal, A. Gehrmann-De Ridder and M. Ritzmann, *Antenna subtraction at NNLO with hadronic initial states: double real radiation for initial-initial configurations with two quark flavours*, *JHEP* **02** (2011) 098, [1011.6631].

[77] T. Gehrmann and P. F. Monni, *Antenna subtraction at NNLO with hadronic initial states: real-virtual initial-initial configurations*, *JHEP* **12** (2011) 049, [1107.4037].

[78] A. Gehrmann-De Ridder, T. Gehrmann and M. Ritzmann, *Antenna subtraction at NNLO with hadronic initial states: double real initial-initial configurations*, *JHEP* **10** (2012) 047, [1207.5779].

[79] A. Gehrmann-De Ridder, T. Gehrmann and E. W. N. Glover, *Quark-gluon antenna functions from neutralino decay*, *Phys. Lett. B* **612** (2005) 36–48, [hep-ph/0501291].

[80] A. Gehrmann-De Ridder, T. Gehrmann and E. W. N. Glover, *Gluon-gluon antenna functions from Higgs boson decay*, *Phys. Lett. B* **612** (2005) 49–60, [hep-ph/0502110].

[81] A. Buckley et al., *General-purpose event generators for LHC physics*, *Phys. Rept.* **504** (2011) 145–233, [1101.2599].

[82] M. Bahr et al., *Herwig++ Physics and Manual*, *Eur. Phys. J. C* **58** (2008) 639–707, [0803.0883].

[83] J. Bellm et al., *Herwig 7.0/Herwig++ 3.0 release note*, *Eur. Phys. J. C* **76** (2016) 196, [1512.01178].

[84] T. Sjostrand, S. Mrenna and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026, [hep-ph/0603175].

[85] C. Bierlich et al., *A comprehensive guide to the physics and usage of PYTHIA 8.3*, 2203.11601.

[86] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert et al., *Event generation with SHERPA 1.1*, *JHEP* **02** (2009) 007, [0811.4622].

[87] Sherpa collaboration, E. Bothmann et al., *Event Generation with Sherpa 2.2*, *SciPost Phys.* **7** (2019) 034, [1905.09127].

[88] F. James, *Monte Carlo Theory and Practice*, *Rept. Prog. Phys.* **43** (1980) 1145.

[89] R. Kleiss, W. J. Stirling and S. D. Ellis, *A New Monte Carlo Treatment of Multiparticle Phase Space at High-energies*, *Comput. Phys. Commun.* **40** (1986) 359.

[90] S. Plätzer, *RAMBO on diet*, 1308.2922.

[91] P. D. Draggiotis, A. van Hameren and R. Kleiss, *SARGE: An Algorithm for generating QCD antennas*, *Phys. Lett. B* **483** (2000) 124–130, [hep-ph/0004047].

[92] A. van Hameren and C. G. Papadopoulos, *A Hierarchical phase space generator for QCD antenna structures*, *Eur. Phys. J. C* **25** (2002) 563–574, [hep-ph/0204055].

[93] G. P. Lepage, *A New Algorithm for Adaptive Multidimensional Integration*, *J. Comput. Phys.* **27** (1978) 192.

[94] G. P. Lepage, *Adaptive multidimensional integration: VEGAS enhanced*, *J. Comput. Phys.* **439** (2021) 110386, [2009.05112].

[95] R. Kleiss and R. Pittau, *Weight optimization in multichannel Monte Carlo*, *Comput. Phys. Commun.* **83** (1994) 141–146, [hep-ph/9405257].

[96] T. Ohl, *Vegas revisited: Adaptive Monte Carlo integration beyond factorization*, *Comput. Phys. Commun.* **120** (1999) 13–19, [hep-ph/9806432].

[97] T. Plehn, *Lectures on LHC Physics*, *Lect. Notes Phys.* **844** (2012) 1–193, [0910.4182].

[98] S. Höche, S. Prestel and H. Schulz, *Simulation of Vector Boson Plus Many Jet Final States at the High Luminosity LHC*, *Phys. Rev. D* **100** (2019) 014024, [1905.05120].

[99] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event Generation with Normalizing Flows*, *Phys. Rev. D* **101** (2020) 076002, [2001.10028].

[100] S. Jadach, *Foam: Multidimensional general purpose Monte Carlo generator with selfadapting symplectic grid*, *Comput. Phys. Commun.* **130** (2000) 244–259, [physics/9910004].

[101] S. Jadach, *Foam: A General purpose cellular Monte Carlo event generator*, *Comput. Phys. Commun.* **152** (2003) 55–100, [physics/0203033].

[102] HSF Physics Event Generator WG collaboration, S. Amoroso et al., *Challenges in Monte Carlo Event Generator Software for High-Luminosity LHC*, *Comput. Softw. Big Sci.* **5** (2021) 12, [2004.13687].

[103] K. Danziger, T. Janßen, S. Schumann and F. Siegert, *Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates*, *SciPost Phys.* **12** (2022) 164, [2109.11964].

[104] M. L. Mangano, M. Moretti, F. Piccinini, R. Pittau and A. D. Polosa, *ALPGEN, a generator for hard multiparton processes in hadronic collisions*, *JHEP* **07** (2003) 001, [hep-ph/0206293].

[105] F. Krauss, R. Kuhn and G. Soff, *AMEGIC++ 1.0: A Matrix element generator in C++*, *JHEP* **02** (2002) 044, [hep-ph/0109036].

[106] T. Gleisberg and S. Hoeche, *Comix, a new matrix element generator*, *JHEP* **12** (2008) 039, [0808.3674].

[107] A. Cafarella, C. G. Papadopoulos and M. Worek, *Helac-Phegas: A Generator for all parton level processes*, *Comput. Phys. Commun.* **180** (2009) 1941–1955, [0710.2427].

[108] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer and T. Stelzer, *MadGraph 5 : Going Beyond*, *JHEP* **06** (2011) 128, [1106.0522].

[109] W. Kilian, T. Ohl and J. Reuter, *WHIZARD: Simulating Multi-Particle Processes at LHC and ILC*, *Eur. Phys. J. C* **71** (2011) 1742, [0708.4233].

[110] C. F. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, H. Ita et al., *An Automated Implementation of On-Shell Methods for One-Loop Amplitudes*, *Phys. Rev. D* **78** (2008) 036003, [0803.4180].

[111] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, G. Ossola et al., *Automated One-Loop Calculations with GoSam*, *Eur. Phys. J. C* **72** (2012) 1889, [1111.2034].

[112] G. Cullen et al., *GOSAM-2.0: a tool for automated one-loop calculations within the Standard Model and beyond*, *Eur. Phys. J. C* **74** (2014) 3001, [1404.7096].

[113] G. Bevilacqua, M. Czakon, M. V. Garzelli, A. van Hameren, A. Kardos, C. G. Papadopoulos et al., *HELAC-NLO*, *Comput. Phys. Commun.* **184** (2013) 986–997, [1110.1499].

[114] V. Hirschi, R. Frederix, S. Frixione, M. V. Garzelli, F. Maltoni and R. Pittau, *Automation of one-loop QCD corrections*, *JHEP* **05** (2011) 044, [1103.0621].

[115] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079, [1405.0301].

[116] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H. S. Shao and M. Zaro, *The automation of next-to-leading order electroweak calculations*, *JHEP* **07** (2018) 185, [1804.10017].

[117] J. M. Campbell and R. K. Ellis, *An Update on vector boson pair production at hadron colliders*, *Phys. Rev. D* **60** (1999) 113006, [hep-ph/9905386].

[118] J. M. Campbell, S. Höche and C. T. Preuss, *Accelerating LHC phenomenology with analytic one-loop amplitudes: A C++ interface to MCFM*, *Eur. Phys. J. C* **81** (2021) 1117, [2107.04472].

[119] S. Badger, B. Biedermann, P. Uwer and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, *Comput. Phys. Commun.* **184** (2013) 1981–1998, [1209.0100].

[120] F. Cascioli, P. Maierhofer and S. Pozzorini, *Scattering Amplitudes with Open Loops*, *Phys. Rev. Lett.* **108** (2012) 111601, [1111.5206].

[121] F. Buccioni, J.-N. Lang, J. M. Lindert, P. Maierhöfer, S. Pozzorini, H. Zhang et al., *OpenLoops 2*, *Eur. Phys. J. C* **79** (2019) 866, [1907.13071].

[122] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf and S. Uccirati, *RECOLA: REcursive Computation of One-Loop Amplitudes*, *Comput. Phys. Commun.* **214** (2017) 140–173, [1605.01090].

[123] A. Denner, J.-N. Lang and S. Uccirati, *Recola2: REcursive Computation of One-Loop Amplitudes 2*, *Comput. Phys. Commun.* **224** (2018) 346–361, [1711.07388].

[124] T. Binoth et al., *A Proposal for a Standard Interface between Monte Carlo Tools and One-Loop Programs*, *Comput. Phys. Commun.* **181** (2010) 1612–1622, [1001.1307].

[125] S. Alioli et al., *Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and one-loop programs*, *Comput. Phys. Commun.* **185** (2014) 560–571, [1308.3462].

[126] J. Campbell, J. Huston and F. Krauss, *The Black Book of Quantum Chromodynamics: A Primer for the LHC Era*. Oxford University Press, 12, 2017.

[127] F. Caola, W. Chen, C. Duhr, X. Liu, B. Mistlberger, F. Petriello et al., *The Path forward to N^3LO*, in *2022 Snowmass Summer Study*, 3, 2022, 2203.06730.

[128] J. M. Campbell et al., *Event Generators for High-Energy Physics Experiments*, in *2022 Snowmass Summer Study*, 3, 2022, 2203.11110.

[129] E. Bothmann, A. Buckley, I. A. Christidi, C. Gütschow, S. Höche, M. Knobbe et al., *Accelerating LHC event generation with simplified pilot runs and fast PDFs*, 2209.00843.

[130] S. Frixione and B. R. Webber, *Matching NLO QCD computations and parton shower simulations*, *JHEP* **06** (2002) 029, [hep-ph/0204244].

[131] S. Frixione, P. Nason and C. Oleari, *Matching NLO QCD computations with Parton Shower simulations: the POWHEG method*, *JHEP* **11** (2007) 070, [0709.2092].

[132] S. Jadach, W. Płaczek, S. Sapeta, A. Siódmok and M. Skrzypek, *Matching NLO QCD with parton shower in Monte Carlo scheme — the KrkNLO method*, *JHEP* **10** (2015) 052, [1503.06849].

[133] S. Catani, F. Krauss, R. Kuhn and B. R. Webber, *QCD matrix elements + parton showers*, *JHEP* **11** (2001) 063, [hep-ph/0109231].

[134] L. Lonnblad and S. Prestel, *Matching Tree-Level Matrix Elements with Interleaved Showers*, *JHEP* **03** (2012) 019, [1109.4829].

[135] B. R. Webber, *A QCD Model for Jet Fragmentation Including Soft Gluon Interference*, *Nucl. Phys. B* **238** (1984) 492–528.

[136] J.-C. Winter, F. Krauss and G. Soff, *A Modified cluster hadronization model*, *Eur. Phys. J. C* **36** (2004) 381–395, [hep-ph/0311085].

[137] B. Andersson, G. Gustafson, G. Ingelman and T. Sjostrand, *Parton Fragmentation and String Dynamics*, *Phys. Rept.* **97** (1983) 31–145.

[138] B. R. Webber, *Fragmentation and hadronization*, *Int. J. Mod. Phys. A* **15S1** (2000) 577–606, [hep-ph/9912292].

[139] M. Cacciari and N. Houdeau, *Meaningful characterisation of perturbative theoretical uncertainties*, *JHEP* **09** (2011) 039, [1105.5152].

[140] "Worldwide lhc computing grid." `https://wlcg.web.cern.ch/`.

[141] I. Zurbano Fernandez et al., *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*, .

[142] A. Krizhevsky, I. Sutskever and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[143] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma et al., *Imagenet large scale visual recognition challenge*, *International journal of computer vision* **115** (2015) 211–252.

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

[145] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[146] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[147] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al., *Pytorch: An imperative style, high-performance deep learning library*, *Advances in neural information processing systems* **32** (2019) .

[148] T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016, DOI.

[149] CMS collaboration, A. M. Sirunyan et al., *Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *JINST* **15** (2020) P10017, [2006.10165].

[150] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, 1707.00028.

[151] M. D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, *SciPost Phys.* **9** (2020) 053, [1810.11509].

[152] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with Neural Importance Sampling*, *SciPost Phys.* **8** (2020) 069, [2001.05478].

[153] B. Stienen and R. Verheyen, *Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows*, *SciPost Phys.* **10** (2021) 038, [2011.13445].

[154] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, *Mach. Learn. Sci. Tech.* **1** (2020) 045023, [2001.05486].

[155] F. Bishara and M. Montull, *(Machine) Learning amplitudes for faster event generation*, 1912.11055.

[156] J. Aylett-Bullock, S. Badger and R. Moodie, *Optimising simulations for diphoton production at hadron colliders using amplitude neural networks*, *JHEP* **08** (2021) 066, [2106.09474].

[157] S. Badger, A. Butter, M. Luchmann, S. Pitz and T. Plehn, *Loop Amplitudes from Precision Networks*, 2206.14831.

[158] G. Bíró, B. Tankó-Bartalis and G. G. Barnaföldi, *Studying Hadronization by Machine Learning Techniques*, 2111.15655.

[159] P. Ilten, T. Menzo, A. Youssef and J. Zupan, *Modeling hadronization using machine learning*, 2203.04983.

[160] A. Ghosh, X. Ju, B. Nachman and A. Siodmok, *Towards a Deep Learning Model for Hadronization*, 2203.12660.

[161] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC*, *JHEP* **08** (2019) 110, [1903.02433].

[162] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC Events*, *SciPost Phys.* **7** (2019) 075, [1907.03764].

[163] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot et al., *Generative Networks for Precision Enthusiasts*, 2110.13632.

[164] J. Arjona Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description*, *J. Phys. Conf. Ser.* **1525** (2020) 012081, [1912.02748].

[165] M. Feickert and B. Nachman, *A Living Review of Machine Learning for Particle Physics*, 2102.02770.

[166] D. Guest, K. Cranmer and D. Whiteson, *Deep Learning and its Application to LHC Physics*, *Ann. Rev. Nucl. Part. Sci.* **68** (2018) 161–181, [1806.11484].

[167] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel et al., *Machine learning at the energy and intensity frontiers of particle physics*, *Nature* **560** (2018) 41–48.

[168] S. Badger et al., *Machine Learning and LHC Event Generation*, 2203.07460.

[169] C. Grojean, A. Paul, Z. Qian and I. Strümke, *Lessons on interpretable machine learning from particle physics*, *Nature Rev. Phys.* **4** (2022) 284–286, [2203.08021].

[170] M. D. Schwartz, *Should artificial intelligence be interpretable to humans?*, .

[171] T. Y. Chen, B. Dey, A. Ghosh, M. Kagan, B. Nord and N. Ramachandra, *Interpretable Uncertainty Quantification in AI for HEP*, in *2022 Snowmass Summer Study*, 8, 2022, 2208.03284, DOI.

[172] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *ICML*, 2010.

[173] D. Hendrycks and K. Gimpel, *Bridging nonlinearities and stochastic regularizers with gaussian error linear units*, *ArXiv* **abs/1606.08415** (2016) .

[174] P. Ramachandran, B. Zoph and Q. V. Le, *Searching for activation functions*, *CoRR* **abs/1710.05941** (2017) , [1710.05941].

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

[176] Z. Lu, H. Pu, F. Wang, Z. Hu and L. Wang, *The expressive power of neural networks: A view from the width*, *Advances in neural information processing systems* **30** (2017) , [1709.02540].

[177] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256, PMLR, 2010, https://proceedings.mlr.press/v9/glorot10a.html.

[178] K. He, X. Zhang, S. Ren and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, *IEEE International Conference on Computer Vision (ICCV 2015)* **1502** (02, 2015) .

[179] J. Kiefer and J. Wolfowitz, *Stochastic estimation of the maximum of a regression function*, *The Annals of Mathematical Statistics* **23** (09, 1952) .

[180] G. Hinton, N. Srivastava and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf, 2012.

[181] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 12, 2014, 1412.6980.

[182] S. Ruder, *An overview of gradient descent optimization algorithms*, 1609.04747.

[183] T. Yu and H. Zhu, *Hyper-parameter optimization: A review of algorithms and applications*, 2003.05689.

[184] J. Bergstra, D. Yamins and D. Cox, *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures*, in *International conference on machine learning*, pp. 115–123, PMLR, 2013, https://proceedings.mlr.press/v28/bergstra13.html.

[185] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins and D. Cox, *Hyperopt: A python library for model selection and hyperparameter optimization*, *Computational Science & Discovery* **8** (07, 2015) 014008.

[186] S. Bubeck, T. Wang and N. Viswanathan, *Multiple identifications in multi-armed bandits*, in *Proceedings of the 30th International Conference on Machine Learning*, PMLR, 2013, https://proceedings.mlr.press/v28/bubeck13.html.

[187] K. Jamieson and A. Talwalkar, *Non-stochastic best arm identification and hyperparameter optimization*, in *Artificial intelligence and statistics*, pp. 240–248, PMLR, 2016, https://proceedings.mlr.press/v51/jun16.html.

[188] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization*, *Journal of Machine Learning Research* **18** (04, 2018) 1–52.

[189] S. Falkner, A. Klein and F. Hutter, *Bohb: Robust and efficient hyperparameter optimization at scale*, in *International Conference on Machine Learning*, pp. 1437–1446, PMLR, 2018, 1807.01774.

[190] S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz De Austri and J. Tattersall, *DeepXS: Fast approximation of MSSM electroweak cross sections at NLO*, *Eur. Phys. J. C* **80** (2020) 12, [1810.08312].

[191] A. Buckley, A. Kvellestad, A. Raklev, P. Scott, J. V. Sparre, J. Van Den Abeele et al., *Xsec: the cross-section evaluation code*, *Eur. Phys. J. C* **80** (2020) 1106, [2006.16273].

[192] A. Alnuqaydan, S. Gleyzer and H. Prosper, *SYMBA: Symbolic Computation of Squared Amplitudes in High Energy Physics with Machine Learning*, 2206.08901.

[193] R. Winterhalder, V. Magerya, E. Villa, S. P. Jones, M. Kerner, A. Butter et al., *Targeting multi-loop integrals with neural networks*, *SciPost Phys.* **12** (2022) 129, [2112.09145].

[194] A. Dersy, M. D. Schwartz and X. Zhang, *Simplifying Polylogarithms with Machine Learning*, 2206.04115.

[195] H. Truong, “Fame.” https://github.com/htruong0/fame, 2021.

[196] A. Bassetto, M. Ciafaloni and G. Marchesini, *Jet structure and infrared sensitive quantities in perturbative qcd*, *Physics Reports* **100** (1983) 201–272.

[197] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896, [1111.6097].

[198] N. Dawe, E. Rodrigues, H. Schreiner, B. Ostdiek, D. Kalinkin, M. R. et al., *scikit-hep/pyjet: Version 1.8.2*, Jan., 2021. 10.5281/zenodo.4446849.

[199] S. Catani, Y. L. Dokshitzer, M. Olsson, G. Turnock and B. R. Webber, *New clustering algorithm for multi - jet cross-sections in e+ e- annihilation*, *Phys. Lett. B* **269** (1991) 432–438.

[200] F. Chollet et al., “Keras.” `https://keras.io`, 2015.

[201] A. Coccaro, M. Pierini, L. Silvestrini and R. Torre, *The dnnlikelihood: enhancing likelihood distribution with deep learning*, *The European Physical Journal C* **80** (Jul, 2020) .

[202] F. Bury and C. Delaere, *Matrix element regression with deep neural networks — breaking the cpu barrier*, *Journal of High Energy Physics* **2021** (Apr, 2021) .

[203] J. Aylett-Bullock, “n3jet.” `https://github.com/JosephPB/n3jet`, 2020.

[204] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad et al., *Deep learning scaling is predictable, empirically*, 2017.

[205] K. Xu, M. Zhang, J. Li, S. S. Du, K. ichi Kawarabayashi and S. Jegelka, *How neural networks extrapolate: From feedforward to graph neural networks*, 2021.

[206] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child et al., *Scaling laws for neural language models*, 2020.

[207] L. Bottou, F. E. Curtis and J. Nocedal, *Optimization methods for large-scale machine learning*, 2018.

[208] Z.-H. Zhou, J. Wu and W. Tang, *Ensembling neural networks: Many could be better than all*, *Artificial Intelligence* **137** (2002) 239–263.

[209] B. Nachman, *A guide for deploying deep learning in lhc searches: How to achieve optimality and account for uncertainty*, *SciPost Physics* **8** (Jun, 2020) .

[210] P. Baldi, K. Cranmer, T. Faucett, P. Sadowski and D. Whiteson, *Parameterized neural networks for high-energy physics*, *Eur. Phys. J. C* **76** (2016) 235, [1601.07913].

[211] A. Ghosh, B. Nachman and D. Whiteson, *Uncertainty-aware machine learning for high energy physics*, *Phys. Rev. D* **104** (2021) 056026, [2105.08742].

[212] O. R. developers, “Onnx runtime.” https://onnxruntime.ai/, 2022.

[213] A. Faus-Golfe et al., *Accelerators for Electroweak Physics and Higgs Boson Studies*, 2209.05827.

[214] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063, [0802.1189].

[215] NNPDF collaboration, R. D. Ball et al., *Parton distributions for the LHC Run II*, *JHEP* **04** (2015) 040, [1410.8849].