# *Colliding Worlds: Modern Computational Methods for Scattering Amplitude Calculations and Responding to Crisis Situations*

JOSEPH PETER AYLETT-BULLOCK

# Colliding Worlds

*Modern Computational Methods for Scattering Amplitude Calculations and Responding to Crisis Situations*

Joseph Peter Aylett-Bullock

A Thesis presented for the degree of
Doctor of Philosophy

Durham University

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

October 2021

# Colliding Worlds

*Modern Computational Methods for Scattering Amplitude Calculations and Responding to Crisis Situations*

Joseph Peter Aylett-Bullock

Submitted for the degree of Doctor of Philosophy

October 2021

**Abstract:** Precision theoretical predictions for high multiplicity scattering rely on the evaluation of increasingly complicated scattering amplitudes which come with an extremely high CPU cost. For state-of-the-art processes this can cause technical bottlenecks in the production of fully differential distributions. In this thesis we explore the possibility of using neural networks to approximate multi-jet scattering amplitudes and provide efficient inputs for Monte Carlo integration. We begin by focussing on QCD corrections to $e^+e^- \to\leq 5$ jets up to one-loop. We demonstrate reliable interpolation when a series of networks are trained on amplitudes that have been divided into sectors defined by their infrared singularity structure. Complete simulations for one-loop distributions show speed improvements of at least an order of magnitude over standard approaches.

We extend our analysis to the case of loop-induced diphoton production through gluon fusion and develop a realistic simulation method that can be applied to hadron collider observables. Specifically, we present a detailed study for $2 \to 3$ and $2 \to 4$ scattering problems which are extremely relevant for future phenomenological studies and find excellent agreement with amplitudes generated using traditional methods. In order to provide a useable technology, we present an interface with the Sherpa Monte Carlo event generator.

The techniques underlying our machine learning methodology and Monte Carlo event generator simulations are widely applicable in other domains as well. In this thesis we will also discuss the use of machine learning to aid in rapid response to crises situations, and the parallels between multi-particle event generators and multi-agent simulations for modelling the spread of epidemics. In this latter case, we develop a new agent-based model with highly granular resolution and discuss its applications to modelling the spread of COVID-19 in England, and in refugee and internally displaced person settlements to aid data driven decision making.

# 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. This thesis is partly based on joint research as noted below.

- Chapter 5 is based on [7]: S. Badger and **J. Bullock**, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes, JHEP* **06** (2020) 114

- Chapter 6 is based on [8]: **J. Aylett-Bullock**, S. Badger and R. Moodie, *Optimising simulations for diphoton production at hadron colliders using amplitude neural networks, JHEP* (under review)

- Chapter 7 discusses research presented in the following works:

  - [9]: E. Nemni, **J. Bullock**, S. Belabbes and L. Bromley, *Fully convolutional neural network for rapid flood segmentation in synthetic aperture radar imagery, Remote Sensing* **12** (2020)

  - [10]: T. Logar, **J. Bullock**, E. Nemni, L. Bromley, J. A. Quinn and M. Luengo-Oroz, *PulseSatellite: A tool using human-AI feedback loops for satellite image analysis in humanitarian contexts, in 34th AAAI Conference on Artificial Intelligence*, AAAI, 2020

  - [11]: **J. Bullock**, A. Luccioni, K. Hoffman Pham, C. Sin Nga Lam and M. Luengo-Oroz, *Mapping the landscape of artificial intelligence applications* *against COVID-19, Journal of Artificial Intelligence Research* **69** (2020) 807–845

- [12]: M. Luengo-Oroz, K. Hoffmann Pham, **J. Bullock**, R. Kirkpatrick, A. Luccioni, S. Rubel et al., *Artificial intelligence cooperation to support the global response to COVID-19, Nature Machine Intelligence* **2** (2020) 295–297
- [13]: A. Luccioni, **J. Bullock**, K. Hoffman Pham, C. Sin Nga Lam and M. Luengo-Oroz, *Considerations, Good Practices, Risks and Pitfalls in Developing AI Solutions Against COVID-19, in Harvard CRCS Workshop on AI for Social Good*, 2020
- [14]: M. Luengo-Oroz, **J. Bullock**, K. Hoffman Pham, C. Sin Nga Lam and A. Luccioni, *From artificial intelligence bias to inequality in the time of covid-19, IEEE Technology and Society Magazine* **40** (2021) 71–79
- [15]: **J. Aylett-Bullock**, C. Cuesta-Lázaro, 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
- [16]: **J. Aylett-Bullock**, C. Cuesta-Lázaro, A. Quera-Bofarull, A. Katta, K. Hoffman Pham, B. Hoover et al., *Operational response simulation tool for epidemics within refugee and IDP settlements, medRxiv* (2021)

The relevant software developed during this thesis has been made available in the following locations:

- The `n3jet` package used in Chapters 5 and 6: `https://github.com/JosephPB/n3jet`
- Specific code accompanying Chapter 6: `https://github.com/JosephPB/n3jet_diphoton`
- Specific code for flood mapping discussed in Chapter 7: `https://github.com/UNITAR-UNOSAT/UNOSAT-AI-Based-Rapid-Mapping-Service`

- The JUNE package discussed in Chapter 7: `https://github.com/IDAS-Durham/JUNE`
- Specific code for adapting JUNE to the refugee and IDP settlements discussed in Chapter 7: `https://github.com/UNGlobalPulse/UNGP-settlement-modelling`

The following pieces of work have been conducted during my doctoral studies but are not included directly in this thesis:

- [17]: **J. Bullock**, C. Cuesta-Lázaro and A. Quera-Bofarull, *XNet: a convolutional neural network (CNN) implementation for medical x-ray image segmentation suitable for small datasets, in Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging* (B. Gimi and A. Krol, eds.), vol. 10953, pp. 453 – 463, International Society for Optics and Photonics, SPIE, 2019
- [18]: **J. Bullock** and M. Luengo-Oroz, *Automated Speech Generation from UN General Assembly Statements: Mapping Risks in AI Generated Texts, in International Conference on Machine Learning AI for Social Good Workshop*, ICML, 2019

# Acknowledgements

I would like to firstly thank my supervisor, Simon Badger, for his support and encouragement throughout my doctoral studies and from whom I have learnt an enormous amount. Thanks also to Frank Krauss who has always encouraged me in my various endeavours and who has made possible many of the opportunities afforded to me throughout my studies.

My sincere thanks goes to everyone at the IPPP — the Institute is a special place with brilliant people and a wonderful community. Thank you in particular to those who I have shared an office with over the years — Andrew, Parisa, Christos, Kevin, Maria Laura, Alan, Marian, and Henry — and to all those who have kindly helped with proof reading this thesis — Andrew, Elliott, Francesco, Henry, Hitham, Kevin, Oscar, Parisa and Ryan. Special thanks goes to Trudy Forster and Joanne Bentham who have ensured everything in the IPPP has run smoothly and always been a source of calm and helpfulness, and to Adam Boutcher and Paul Clark for keep the computing infrastructure up and running, even during a pandemic.

Throughout my time at Durham, I have been fortunate enough to be part of another amazing community — St. John's College. The College has been a home to me for many years and I'm indebted to the many brilliant staff who work there and make it possible — David Wilkinson, Mark Ogden, Rebecca Bouveng, Angela Cook, Alan Usher, Sue Hobson, Lynne Ramage, Alison Bradshaw and many others.

Thank you to United Nations Global Pulse (UNGP) for the opportunity of a research Fellowship. The team at UNGP is truly inspiring and it has been a pleasure to work alongside such talented and passionate people. A special thanks to Miguel Luengo-Oroz for his support, guidance, and mentorship over the past few years.

I have also had the pleasure of being a Research Associate at the RiskEcon Lab for Decision Metrics, part of the Courant Institute of Mathematical Sciences at New York University. Thank you to David Mordecai and Samantha Kappagoda for their leadership and mentorship, and for making this position possible.

Much of the work presented in this thesis has been done alongside many other brilliant people. Thank you to all of my collaborators for your wisdom, guidance, expertise, and for making the research we've done together a truly enjoyable and exciting experience. I look forward to collaborating with many of you in the future.

I would like to thank my parents, for their constant love, always being there to support me, and for inspiring me from a young age in science. Finally, thank you to my wife, Caragh, who has always been there for me and who has supported me unendingly throughout my doctoral studies. Thank you also for doing me the honour of agreeing to marry me part way through my doctorate, and then marrying me during a pandemic.

The work presented in this thesis has been funded by UKRI-STFC grant number ST/P006744/1, as well as a Fellowship from United Nations Global Pulse. United Nations Global Pulse work is supported by the Government of Sweden, and the William and Flora Hewlett Foundation.

# Chapter 1

# Introduction

The current best working theory to describe fundamental interactions in particle physics is the Standard Model (SM). The theory consists of fermions, the building blocks of matter, and bosons, which mediate interactions. The strength of the SM has been demonstrated on numerous occasions, with one of the most historic being the discovery of the Higgs Boson in 2012 at the ATLAS [19] and CMS [20] experiments at the Large Hadron Collider (LHC). Despite this, there are several long-standing issues which have not been addressed by the SM. These include cosmological measurements suggesting the presence of Dark Matter and Dark Energy, the CP violating asymmetry of the apparent existence of matter over antimatter, and the lack of inclusion of gravity in the theory. Many Beyond the Standard Model (BSM) theories and studies are being explored and proposed to tackle these, although no discovery of physics beyond the SM has been made at the LHC.

The SM is a quantum field theory in which particles are considered to be excitations of their respective fields. The fields in the SM interact under a gauge group which defines the symmetries of the theory

$$SU(3) \times SU(2) \times U(1). \tag{1.0.1}$$

The first term in the expression above encapsulates the interactions of the strong force, mediated by gluons, which is described by the theory of Quantum Chromodynamics

| Gauge Field | $SU(3)$ | $SU(2)$ | $U(1)$ |
|---|---|---|---|
| $A$ | **8** | 0 | 0 |
| $W$ | 0 | **3** | 0 |
| $B$ | 0 | 0 | **1** |

Table 1.1: The number of gauge fields in each group appearing in the Standard Model.

(QCD). The final two terms contain the electroweak sector which is broken by the Higgs mechanism [21–23].

The Lagrangian of a theory, $\mathcal{L}$, describes the interactions of the fields. The mixing of the fields in the expanded Lagrangian determines the rules for the propagators and allowed interactions in the theory — the Feynman rules. In the case of the SM, the gauge group in Equation 1.0.1 determines the number of gauge fields, which are given in Table 1.1. The gluons correspond to the eight gluon fields, $A$, while the photon and $W$ and $Z$-bosons appear as mixtures of the $B$ and $W$ fields after electroweak symmetry breaking has occurred. In this thesis we will primarily focus on the QCD components of the SM Lagrangian. Strong corrections are a crucial part of SM calculations and probing the theory will allow us to better uncover any potential small deviations which may relate to new physics. We will introduce the theory of QCD in more detail in Chapter 2.

To test the SM in practice, colliders such as the LHC accelerate particles to near the speed of light before inducing collisions and measuring the final results. When particles collide, there are many possible outcomes, and the probability that a particular outcome occurs is given by the cross-section [1]

$$\sigma \sim \int \mathrm{d}\Phi_{n-2} |\mathcal{M}(p_1 p_2 \to p_3 ... p_n)|^2. \tag{1.0.2}$$

The probability that a collision process occurs for a specific set of initial and final state particle momenta is given by it's squared matrix element, $|\mathcal{M}|^2$, (see Section 2.2) which is calculated by following the Feynman rules of the theory defined by

[1]Note that Equation 1.0.2 does not explicitly include Parton Distribution Function contributions and is designed to illustrate the structure of the partonic cross-section.

the Lagrangian (see Appendix A). For two incoming particles of a given momentum configuration, the (partonic) cross-section is then determined by integrating over all possible outgoing momentum configurations, $\mathrm{d}\Phi_{n-2}$ (see Section 3.1 for a more complete mathematical discussion). A schematic of this approach is given in Figure 1.1.

$$\mathcal{L} \xrightarrow{\text{Feynman Rules}} |\mathcal{M}|^2 \xrightarrow{\text{Integrate}} \sigma$$

Figure 1.1: Simple workflow describing the process of obtaining the cross-section of an event: the Lagrangian defines the theory; Feynman rules describing the propagator and interaction terms of the theory can be obtained from the Lagrangian; these are combined to calculate the matrix element; the matrix elements are integrated over all allowed final state momenta configurations to obtain the cross-section.

In SM calculations at hadron colliders, strongly interacting radiation modelled by QCD dominates — gluons can be radiated and interact with any coloured particle, including themselves, and the QCD coupling is significantly larger than the QED equivalent. QCD is an *asymptotically free* theory, meaning that the coupling decreases as the energy increases. Around the mass of the $Z$-boson the coupling takes a value of approximately $\alpha_s(M_Z) = 0.118$, and so we can treat QCD *perturbatively*, with this approximation becoming increasingly precise as the energy grows. We will discuss these properties in more detail in Section 2.4.

Experimental methods in particle physics are now able to probe interactions at high energies — at the time of writing the LHC is running at $\sqrt{s_{\text{com}}} = 13$ TeV with plans to increase this, and more importantly the luminosity, through the High Luminosity LHC (HL-LHC) programme. Measurements of many observables at the LHC are already being made at the percent level, and with these upgrades experimental findings will become increasingly precise. To obtain this level of precision in calculations, one would expect to require at least three orders in the perturbative expansion:

leading order (LO), next-to-leading order (NLO) and next-to-next-to-leading order (NNLO). However, calculating these higher order terms can require increasingly sophisticated techniques to keep the computational time for cross-section calculation from being prohibitively expensive.

To simulate collider results, we use Monte Carlo event generators to model different stages of the physical collision process and the subsequent multi-particle interactions. Event generators allow for the simulation of many possible particle interactions, and combinations of interactions, to calculate physical observables. However, the high luminosity, and therefore increased precision, now being reached by collider experiments has put a strain on the theoretical calculations. There are many bottlenecks to event generation simulations at the cutting edge of the precision frontier, such as efficient phase-space sampling for cross-section calculation, and parton shower matching algorithms for higher order corrections (see Chapter 3). One of the most significant is the computation of matrix elements for higher order terms in perturbative QCD which require repeated evaluation for the calculation of observables. While numerical implementations for these matrix elements exist, full cross-section calculations can be computationally demanding. New techniques to reach high precision theoretical QCD calculations are needed to ensure theoretical calculations can keep up with experimental development.

Machine learning (ML) techniques have been shown to have many applications in particle physics, including regression, classification and generative tasks. ML methods have the potential to be used for complex analysis procedures, incorporating a greater amount of data, as well as speeding up existing time-consuming calculations. Care must be taken since, due to their complex structures, machine learning models can be comparatively opaque in contrast to more classical techniques, and uncertainties can be introduced at multiple stages of their development. In this thesis we will be concerned with ways in which ML techniques can be used to address some of the event generation bottlenecks, and specifically focus on methods to approximate high dimensional functions for use in matrix element calculations at high precision and their interfacing with event generators.

In this thesis we will also briefly introduce some of the additional work conducted alongside the particle physics research mentioned above. Many of the techniques discussed and developed in this thesis originated from cross-disciplinary efforts and are widely applicable to multiple fields. As an example, machine learning has the potential to alleviate much of the manual analysis required to generate data and insights vital for informing relief efforts in humanitarian crises. We will discuss ML computer vision techniques for mapping refugee and internally displaced person (IDP) settlements and flood response, as well as various ways ML has been used to address the COVID-19 pandemic. We will also discuss alternative ways to use some of the techniques employed in Monte Carlo event generators. Event generators can be thought of as multi-agent simulations using probabilistic modelling to emulate particle interactions. The fundamental techniques underpinning event generators can therefore be applied to epidemic models which simulate the movement and interactions of individuals in a population. We discuss this in the context of modelling the spread of COVID-19 through the population of England, as well as in the context of refugee and IDP settlements where populations are particularly at risk of rapid disease spread given their dense living conditions and limited access to healthcare.

This thesis is divided into the following Chapters. In Chapter 2 we introduce the theory of QCD, and discuss the origins and implications of perturbative QCD as well as some of the theoretical challenges. Chapter 3 will focus on introducing the relevant components of physical observable calculations in the context of Monte Carlo integration and event generators, and how these tools are used to make theoretical predictions. A brief introduction to ML techniques in the context of phenomenological studies of particle physics will be given in Chapter 4. In Chapters 5 and 6 we will describe how ML techniques can be used for matrix element calculations in the case of $e^+e^-$ collisions and hadronic collisions, respectively. The latter of these will focus specifically on the gluon-induced diphoton amplitudes which can be particularly challenging to compute at high multiplicity, and the interfacing of these methods with existing event generator technology. Chapter 7 will give a broad overview of some additional work looking at ways to use ML approaches, and techniques employed in Monte Carlo event generators, in the context of crisis response and epidemic modelling. Finally, Chapter 8 will close with a discussion of how these various methods and new approaches can be combined to enable high precision theoretical predictions for collider experiments and how interdisciplinary research can help further the all participating disciplines.

# Chapter 2

# Introduction to QCD

In this chapter we will provide a brief introduction to the theory of QCD, with the objective of providing sufficient detail regarding the various mathematical tools employed throughout this thesis, as well as some introductory background on their origins. The focus here will be partonic level calculations in QCD, which will later be extended to the broader context of hadronic calculations in Chapter 3. This Chapter is based on and inspired by multiple sources [24–29].

## 2.1 The Theory of QCD

The theory of QCD governs the interactions of the strong force, mediated by gluons. In full generality, QCD can be considered as an $SU(N_c)$ gauge field theory, where we refer to $N_c$ as the number of colours in the theory (in the Standard Model $N_c = 3$). QCD is governed by the Lagrangian

$$\mathcal{L} = -\frac{1}{4}F^{a,\mu\nu}F^{a}_{\mu\nu} + \bar{\psi}^{i}(i\not{D} - m)\psi^{j} + \mathcal{L}_{gauge} + \mathcal{L}_{CP\,violating}. \tag{2.1.1}$$

The first term in Equation 2.1.1 describes the Yang-Mills theory [30] and the second, the interactions between the quark fields, $\psi$, with corresponding mass $m$ (where the flavour indices on the quark fields and masses have been suppressed). The additional terms not detailed here still play important roles — the gauge fixing terms are necessary due to an overcounting in the number of degrees of freedom when quantising the theory, which requires a gauge choice to resolve, and the CP violating term is permitted since it is gauge invariant and renormalisable. The Dirac-slashed notation is defined as: $\not{p} = \gamma_\mu p^\mu$.

The quarks transform in the fundamental representation and the gluons in the adjoint, where the gluon fields, $A^a_\mu$, are apparent in the covariant derivative

$$D^\mu = \partial^\mu - ig_s A^{a,\mu} t^a, \tag{2.1.2}$$

and the gluon field strength tensor, $F^{a,\mu\nu}$, is defined as

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g_s f^{abc} A^b_\mu A^c_\nu. \tag{2.1.3}$$

In the above equations, $i, j \in [1, N_c]$ are the fundamental colour indices, Greek indices are Lorentz indices, and $a, b, c \in [1, N_c^2 - 1]$ denote the adjoint indices. [1] Substituting Equations 2.1.2 and 2.1.3 back into the QCD Lagrangian, it is clear that after expansion there will be non-trivial mixing terms including the gluon and quark fields, as well as triple and quadruple gluon fields terms. This will become relevant when we discuss the construction of matrix elements in Section 2.2 and is fundamental to why higher order QCD corrections can quickly become complex, with a rapidly growing number of Feynman diagrams at each subsequent order.

The gauge group generators appearing in Equation 2.1.2 are related to the structure constant, $f^{abc}$, which defines the group algebra

$$[t^a, t^b] = if^{abc} t^c. \tag{2.1.4}$$

Throughout this thesis we will be concerned with perturbative QCD. Specifically, we will focus on the calculation of matrix elements (see Section 2.2), which we will calculate by expanding around the all important coupling

$$\alpha_s = \frac{g_s^2}{4\pi}, \tag{2.1.5}$$

[1] In the case of $N_c = 3$, the adjoint representation of the gluons gives rise to the eight gluon fields in Table 1.1.

where $g_s$ is the strong coupling. The ability to expand perturbatively relies on the assumption that $\alpha_s \ll 1$, which turns out to be a valid assumption, particularly at the increasingly high energies achieved at particle colliders. We will discuss this in more detail in Section 2.4.

## 2.2 Matrix Elements

In scattering processes the evolution of incoming initial states to outgoing final states is defined through the $\mathcal{S}$-matrix, which encodes a sequence of unitary operations

$$|\text{out}\rangle = \mathcal{S}\,|\text{in}\rangle\,, \tag{2.2.1}$$

where the 'in' and 'out' Fock spaces are isomorphic. In the case where the particles do not interact, the matrix is simply the identity, meaning we can separate out the interacting parts

$$\mathcal{S} = \mathbf{1} + i\mathcal{T}. \tag{2.2.2}$$

In a scattering process involving $n$ particles, we can now write

$$\langle\text{out}|\,i\mathcal{T}\,|\text{in}\rangle = (2\pi)^4\delta^{(4)}(P_{\text{in}} - P_{\text{out}})\,i\mathcal{M}(p_1, ..., p_m; p_{m+1}, ..., p_n), \tag{2.2.3}$$

where the $\delta$-function ensures momentum conservation between the initial and final states (which include initial state momenta, $P_{\text{in}} = p_1 + ... + p_m$, and final state, $P_{\text{out}} = p_{m+1} + ... + p_n$, respectively), and the process is defined in terms of *matrix elements*, $\mathcal{M}$, otherwise known as *scattering amplitudes.* These matrix elements are fundamental to calculating observables in particle collision processes and their calculation forms the majority of this thesis.

Now that we have defined the matrix elements, we can use Feynman rules to calculate them for each process. The matrix element for a given process is the sum over all Feynman diagrams. The rules for QCD can be derived by expanding the Lagrangian (Equations 2.1.1-2.1.3), computing the propagators, and reading off the coefficients of the interacting field terms. These are given in Appendix A.

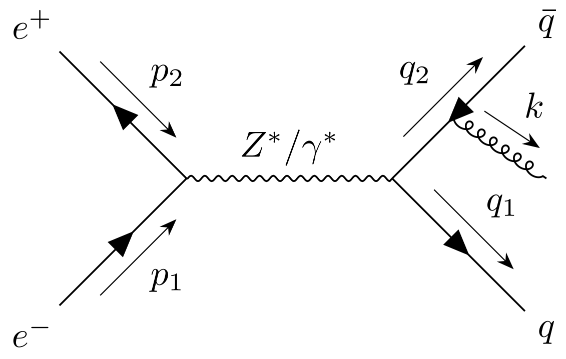


Figure 2.1: A Feynman diagram for the process $e^+e^- \to q\bar{q}g$ where a gluon is emitted from the outgoing antiquark. There is an additional diagram for this process which depicts the gluon emission from the outgoing quark.

As an example, we apply these rules to the calculation of the matrix element for the simple process $e^+e^- \to q\bar{q}g$. Figure 2.1 depicts one of the diagrams contributing to this process, where the gluon is emitted from the outgoing antiquark. The other diagram is the same, but with the gluon emitted from the outgoing quark. The first of these diagrams has a partial matrix element

$$i\mathcal{M}_1 = \left[\bar{u}(q_1)(-ig_eQ_f\gamma^\mu)\frac{i(\not{q}_2 + \not{k})}{(q_2+k)^2}(ig_s t^a_{ij}\not{\epsilon}^*(k))v(q_2)\right] \times \frac{-i}{(p_1+p_2)^2}[\bar{v}(p_2)(ig_e\gamma_\mu)u(p_1)], \quad (2.2.4)$$

where $g_e$ is the QED coupling constant, $Q_f$ is the charge of the quark of flavour $f \in \{b, s, ...\}$, and $u$ ($\bar{u}$) and $\bar{v}$ ($v$) are spinors associated with incoming (outgoing) external spin-1/2 particles and antiparticles respectively, and we have have dropped the Feynman parameter for simplicity. The expression for $i\mathcal{M}_2$, where the gluon is emitted from the quark, follows similarly, and the squared amplitude can be calculated by summing these contributions together and squaring the sum.

Equation 2.2.4 shows how the radiated gluon introduces a term in $\alpha_s$ through the quark-gluon vertex. By adding more gluons we can draw diagrams in increasing powers of $\alpha_s$, and these contribute at higher orders in perturbative QCD (this will be discussed further in Section 2.3.2). For a given set of initial and final state momenta, the squared matrix elements entering into the cross-section calculations (see Equation 1.0.2) are then just real numbers.

## 2.3 Divergent Structures

Feynman rules allow for the construction of analytic formulae for describing scattering processes from simple diagrammatic representations. It is clear, however, that certain divergences will arise from the integral expressions when we sum over all possible momentum states. These divergences are divided into two categories: ultraviolet (UV) divergences, which occur in the large limits of the propagator integrals, and infrared (IR) divergences, which arise from the low energy limits. The former can be resolved through the renormalisation techniques, which appear as corrections in the QCD Lagrangian, while the latter require a more involved procedure.

### 2.3.1 Ultraviolet Divergences

UV divergences arise from the high energy limits of integrals such as

$$\int_0^\Lambda \frac{\mathrm{d}^4 l}{(l^2)(l+k)^2} \sim \log(\Lambda) \xrightarrow[\Lambda\to\infty]{} \infty, \tag{2.3.1}$$

which appear in loop integral expressions.

Renormalisation allows us to resolve these divergences at each loop order by introducing *counter-terms* designed to explicitly cancel such divergent structures. These counter-terms are only allowed after careful construction to ensure that gauge invariance and unitarity are preserved. The counter-terms appear as rescalings of the fields and constants in the Lagrangian (Equation 2.1.1) such that the renormalised Lagrangian can be written as

$$\begin{aligned}\mathcal{L} =& -\frac{1}{4}F^{\mu\nu}F_{\mu\nu} + \bar{\psi}(i\not{D} - m)\psi \\ &+ \bar{\psi}(i\delta_2\not{\partial} - \delta_m)\psi - \frac{1}{4}\delta_3(\partial_\mu A^a_\nu - \partial_\nu A^a_\mu) \\ &+ g_s\delta_1\bar{\psi}t^a\not{A}^a\psi - g_s\delta_1^{3g}f^{abc}\partial_\mu A^a_\nu A^{b,\mu}A^{c,\nu} - g_s\delta_1^{4g}f^{abc}f^{cde}A^b_\mu A^b_\nu A^{d,\mu}A^{e,\nu}.\end{aligned} \tag{2.3.2}$$

In Equation 2.3.2, the fields have all been renormalised relative to the 'bare' form appearing in Equation 2.1.1, and the counter-terms are represented by $\delta_i$. The counter-terms are evaluated at each order in perturbation theory. These new terms will give rise to additional Feynman rules to cancel the divergences. Given this, and that the counter-terms enter the renormalised Lagrangian through parameter rescalings, renormalisation is only possible when the number of divergent processes is less than or equal to the number of parameters in the Lagrangian which can be rescaled. A more complete introduction to renormalisation in general is given in Part II of [24], and in Section 3 of [28] in the context of QCD.

When performing calculations which require the evaluation of divergent loop integrals, such as in Equation 2.3.1, and the subtraction of counter-terms, we need a regulator to control the divergence of the individual terms. A naive implementation is to use the $\Lambda$ in Equation 2.3.1 as a UV cut-off scale, however, the most common method is *dimensional regularisation* in which the integrals are calculated in $d = 4 - 2\epsilon$ dimensions resulting in terms of the form $1/\epsilon$. The integrals are finite in $d$ dimensions and will remain so as $\epsilon \to 0$ once all divergent integrals have been calculated and counter-terms subtracted. We will assume the use of dimensional regularisation throughout this thesis and more information can be found in Chapter 7 [24].

The counter-terms introduce a *renormalisation scale*, $\mu_R$, which is a free parameter in the case of massless QCD. In particular, under renormalisation procedure, the QCD coupling becomes a function of this scale choice $\alpha_s(\mu_R^2)$, the consequences of which will be discussed in more detail in Section 2.4.

### 2.3.2 Infrared Divergences

IR divergences occur in the low energy limits, $p \to 0$, which can occur in two settings: loop integrals (virtual IR divergences) and divergences resulting from soft and collinear emissions (real IR divergences).

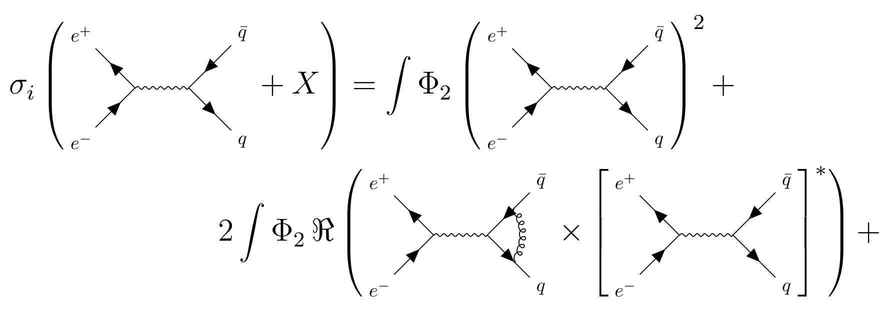
Virtual IR divergences are apparent in the low energy limits of expressions similar to Equation 2.3.1

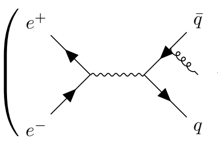
$$\int_{\Lambda_{\mathrm{IR}}}^{\Lambda_{\mathrm{UV}}} \frac{\mathrm{d}^d k}{k^2} \xrightarrow[\Lambda_{\mathrm{IR}}\to 0]{} \infty. \tag{2.3.3}$$

Real IR divergences occur when invariant mass terms approach zero while appearing in the denominators of matrix elements (see the Parke-Taylor formula for a particularly apparent example of this [31]). To achieve this, two outgoing particles can either go collinear ($\theta_{12} \to 0$, where $p_1$ and $p_2$ are the 4-momenta of the two collinear particles), or an outgoing gluon can go soft ($E_g \to 0$). These can be observed by expanding out the denominator in the gluon propagator term in Equation 2.2.4, when the antiquark and gluon can go collinear, or the outgoing gluon can go soft. We expand on this in Chapter 5.

As in the case of UV divergences, physical observables do not contain divergences and so we must also find a way to cancel the IR divergent structures. The UV divergences were cancelled through the introduction of counter-terms in the renormalisation procedure; however, the IR divergences are more challenging. This is because the Kinoshita-Lee-Nauenberg (KLN) [32,33] and Bloch-Nordsieck (BN) [34] theorems ensure that the Standard Model is perturbatively infrared finite through the cancellation of virtual IR divergences with those from the real radiation contributions. This exact cancellation can be observed upon integration over the whole phase-space of the final state particles.

To achieve this cancellation, at any given order in perturbation theory, we must include both the virtual corrections and the equivalent unresolved real emissions to calculate the *inclusive* cross-section. A schematic of this based on the $e^+e^- \to q\bar{q}$ process is as follows:



$$\int \Phi_3 \Bigg($$


$$+ q \leftrightarrow \bar{q} \Bigg)^2 + \mathcal{O}(\alpha_s^2), \quad (2.3.4)$$

which produces an infrared finite cross-section upon cancellation of these singular structures. The first integral of Equation 2.3.4 contains the Born approximation tree-level (i.e. no loops) term, the second and third integrals contain the order $\alpha_s$ corrections. In practice, this cancellation is made possible through the factorisation of scattering amplitudes in which both the virtual and real emission divergences are proportional to the tree-level amplitude.

In reality, integrating over the entire phase-space of the final state particles becomes extremely challenging to solve analytically at higher multiplicity, requiring the use of Monte Carlo integration techniques (as will be discussed in Chapter 3). The use of such numerical techniques means that another approach to the cancelation of infrared terms must be used — infrared subtraction. Commonly employed methods include Catani-Seymour (CS) subtraction [35,36], and Frixione, Kunszt and Signer (FKS) subtraction [37,38] (the concept of which will be briefly discussed in Chapter 5 and Appendix C, but will not be the focus of this thesis). In this thesis, we will not perform full real subtraction when integrating over phase-space, but rather focus on the integration of leading order and one-loop virtual corrections. To ensure the real IR divergences do not grow too large, we will introduce commonly used global phase-space cuts (see Chapters 5 and 6).

## 2.4 Running Coupling

So far we have discussed the theory of QCD in generality, how to compute matrix elements from Feynman rules, as well as some of the subtleties of the theory such as how to manage complex divergent structures which appear because of these very same Feynman rules. In Chapter 1 and Section 2.1, we introduced the notion of perturbative QCD and have made multiple references to this throughout the Chapter.

We will now turn our attention to understanding what enables this perturbative approach — the running of the coupling, $\alpha_s$.

In Section 2.3, we introduced the renormalisation scale, $\mu_R$, which can be arbitrarily chosen, meaning the theory should remain independent of the choice of scale. The dependence of the coupling on this scale is governed by the so-called $\beta$-function. By solving the Callan-Symanzik equation [39, 40], we find

$$\beta(\alpha_s) = \frac{\mathrm{d}\alpha_s}{\mathrm{d}\log(\mu_R^2)}. \tag{2.4.1}$$

which demonstrates that the running of the rescaled coupling constant, $\alpha_s(\mu_R^2)$, is logarithmic with the renormalisation scale (see Section 3 in [41] for details). Expanding the $\beta$-function in $\alpha_s$ gives

$$\beta(\alpha_s) = -\alpha_s^2(b_0 + b_1\alpha_s + b_2\alpha_s^2 + ...). \tag{2.4.2}$$

In QCD, these coupling constants appear at each loop order (see Equation 2.3.4) and so we can calculate coefficients of the $\beta$-function perturbatively in $\alpha_s$ by evaluating the coefficients at each order. At leading order — i.e. one-loop — we obtain

$$b_0 = \frac{1}{3}(11N_c - 2N_f), \tag{2.4.3}$$

where $N_f$ is the number of quark flavours. In the Standard Model, $N_c = 3$ and $N_f = 6$ and so $b_0$ is positive. This has important ramifications as it means that the theory is *asymptotically free* [42, 43] — i.e. that the coupling becomes increasingly small as the renormalisation energy scale grows. This has also been confirmed experimentally with close agreement between theoretical and experimental results (see Figure 2.2). The asymptotic freedom of QCD is in contrast to QED and means that as we probe higher and higher energies in particle physics experiments, the perturbative nature of QCD becomes more exact. However, as we begin to probe these higher energy scales with higher luminosity, there is an increased requirement for higher order QCD calculations which can become increasingly complex and computationally intensive and is the source of one of the crucial bottlenecks in theoretical calculations.

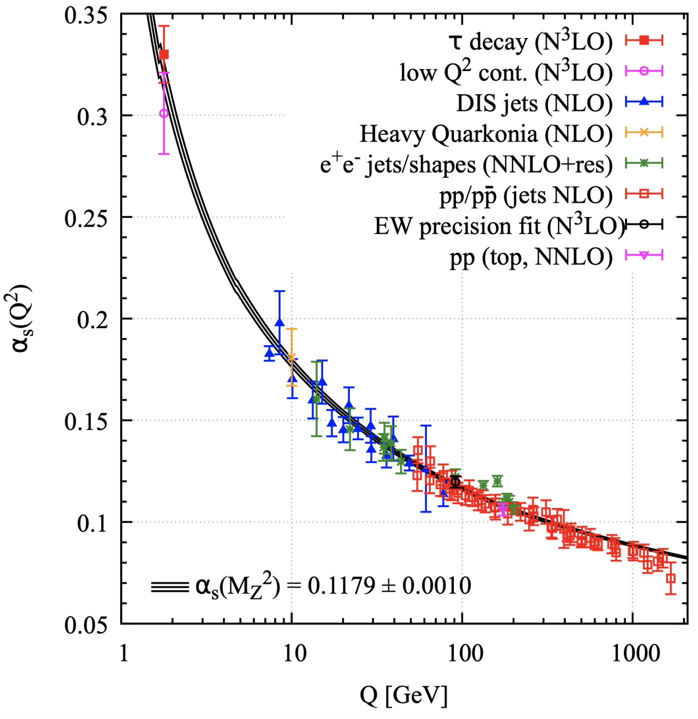


Figure 2.2: The running of $\alpha_s$ as measured by experiments compared with theory. The degree of QCD perturbation theory used to extract $\alpha_s$ is given in brackets. Figure from [1].

In general, the renormalisation scale is chosen to be close to the energy scale of the hard scattering process (see Chapter 3 for more details) as this cancels terms in the calculations which contain ratios of the energy scale to the renormalisation scale. Due to good statistics in its measurement, $\mu_R^2 = M_Z^2$ (where $M_Z$ is the mass of the $Z$-boson) is one of the most common choices. In addition, when performing theoretical calculations, the theoretical uncertainty on the scale choice is given. This uncertainty is calculated by varying the free scales — renormalisation and factorisation scales (see Chapter 3) — by a factor of two in both directions. In perturbative QCD calculations this uncertainty, and therefore the scale dependence, will decrease at higher orders.

It is worth noting that this perturbative approach only works if we can expand around the coupling constant, and therefore that $\alpha_s \ll 1$. This requirement imposes a lower cut-off value, $\Lambda_{\text{QCD}}$, at which the perturbative approach is no longer applicable. This occurs at $\Lambda_{\text{QCD}} \approx 250\,\text{MeV}$ and results in the property of colour confinement. As current particle experiments operate on the energy scales of TeV, $\alpha_s$ is sufficiently small to be able to use perturbative techniques and treat quarks as asymptotically free during hard scattering calculations.

# Chapter 3

# Monte Carlo Event Generators

Now that we have reviewed the underlying theory of QCD, we are able to broaden our discussion to consider how these calculations are used in practice to link theory and experiment. The purpose of this Chapter is to provide a brief introduction to some of the computational techniques for calculating experimental observables. We will begin with a discussion of these observables and then proceed to the automation of their calculation in the context of Monte Carlo event generators. This Chapter is based on and inspired by multiple sources [24, 25, 27, 44].

## 3.1 Measurements and Observables

Experiments such as the LHC have now made the colliding of massive nuclear particles at high energies possible, resulting in many complex interactions. Detectors at these experiments are designed to measure both the energy and direction of the products of collision events. Computational simulations such as event generators are used to reconstruct different stages of the collision processes to match theory to experiment, and vice versa.

One of the most important quantities calculated in quantum field theory is the scattering cross-section. The notion of cross-section was introduced in Chapter 1 with matrix element calculations discussed in Chapter 2. Here we will give a more complete mathematical description.

The cross-section is formally linked to the matrix element of a given process through a factorisation scale, $\mu_F$, which defines the cut-off between long-range hadronic effects and short-range effects which can be calculated using perturbative QCD. [1] The full hadronic cross-section for a collision process of two hadrons, $h_1$ and $h_2$, in the initial state, and some final state composition, $X$, can be calculated using

$$\sigma_h(h_1, h_2 \to X) = \sum_{p_1=\{g,u,d,...\}} \sum_{p_2=\{g,u,d,...\}} \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \\ f_{h_1}^{(p_1)}(x_1, \mu_F^2) f_{h_2}^{(p_2)}(x_2, \mu_F^2) \sigma\left(p_1 p_2 \to X; \mu_F, \mu_R\right), \quad (3.1.1)$$

where $f_h^{(p)}$ are the parton distribution functions (PDFs) which depend on the factorisation scale, and on the momenta fractions, $x$, of the partons, $p$, which make up the hadrons, $h$, (the flavours which are summed over depend on the hadron compositions). At leading order, the PDFs can be interpreted as encoding the probabilities of finding a parton with a given momentum fraction at a certain scale, $\mu_F$, in a hadron. [2] For more details on the PDFs, see Chapter 2 of [25] and Chapter 17 of [24].

The final component of Equation 3.1.1 is the partonic cross-section, $\sigma$, which describes the cross-section at the level of individual partonic constituents of the initial state hadrons

$$\sigma(p_1 p_2 \to p_3 ... p_n) = \frac{\mathbb{S}}{2 s_{\text{com}}} \int \prod_{i=3}^{n} \\ \frac{\mathrm{d}^3 p_i}{2(2\pi)^3 E_i} (2\pi)^4 \delta^{(4)} \left(p_1 + p_2 - \sum_{i=3}^{n} p_i\right) \left|\mathcal{M}(p_1 p_2 \to p_3 ... p_n)\right|^2, \quad (3.1.2)$$

where $\mathbb{S}$ is a symmetry factor to account for symmetries associated with identical

[1]It is common to choose $\mu_F = \mu_R$ and calculate the theoretical scale uncertainty by varying these scales by a factor of two.

[2]It is worth noting, that Equation 3.1.1 does not quite describe the full picture as we have suppressed higher twist terms which encode contributions from interactions e.g. two partons from each hadron interacting. However, considering only the leading twist contributions will be sufficient for the purposes of this thesis.

particles in the final state, and $s_{\text{com}}$ is related to the the partonic centre of mass (c.o.m.) energy through $E_{\text{com}} = \sqrt{s_{\text{com}}}$ and $s_{\text{com}} = (p_1 + p_2)^2$. The matrix elements, $\mathcal{M}$, are calculated as in Section 2.2.

Alongside the cross-section calculation itself, differential distributions, $\mathrm{d}\sigma/\mathrm{d}\mathcal{O}$ (where $\mathcal{O}$ is some variable such as transverse momentum $p_T$), can provide more context since they demonstrate how the cross-section changes over different slices of phase-space. Both cross-section and differential cross-section calculations will be used throughout this thesis when assessing the performance of various machine learning approximations to computationally intensive matrix element calculations in the context of event generation simulations.

## 3.2 Brief Overview of Event Generators

The high level of complexity in calculating observables is due, in part, to the many possible outcomes of particle collisions, and the numerous sub-calculations which need to be completed to arrive at an understanding of one possible outcome. Each of these sub-processes is governed by, often non-trivial, probability distributions which require the evaluation of multiple integral expressions such that all possible outcomes are accounted for.

Many different techniques and computational packages exist to calculate the various stages of a collision event and are brought together in Monte Carlo event generators. There are various flavours of general purpose event generators, such as Pythia [45, 46], Herwig [47–49], Sherpa [2, 50], and MadGraph [51]. In this thesis, we will mainly make use of Sherpa. However, the details covered in this Chapter are broadly applicable to all general purpose methods which we will refer to as simply 'event generators'.

In order to perform the theoretical calculations which can be used to match against experimental observables, and vice versa, event generators break down the collision

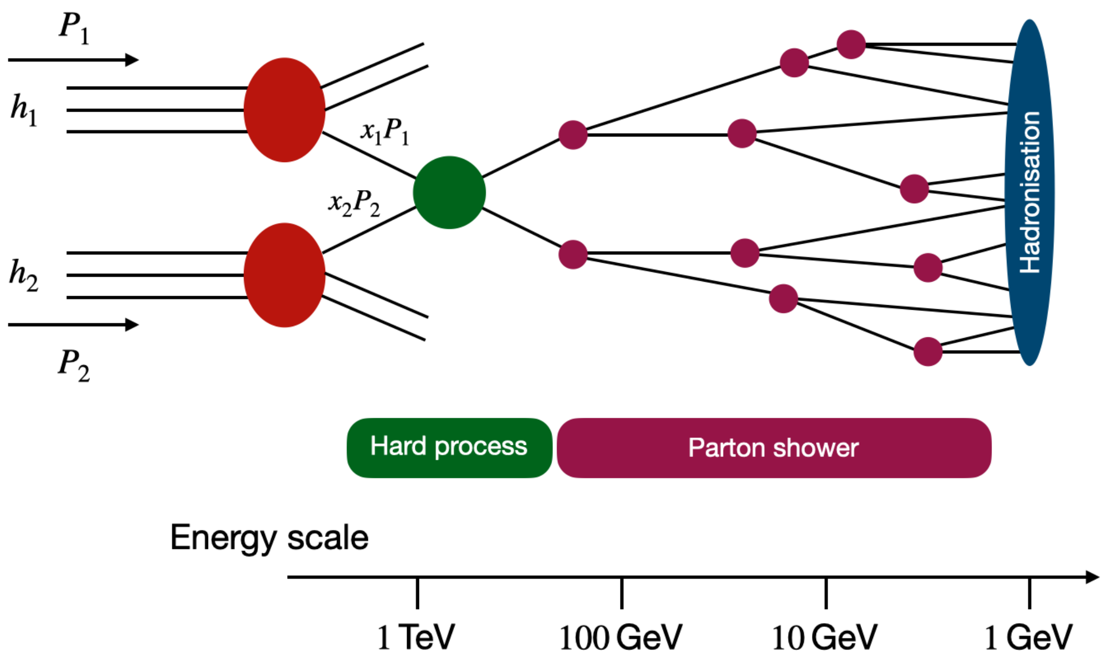


Figure 3.1: Representation of the stages of event generation for hadronic collisions. The incoming hadrons, $h_1$ and $h_2$ have momenta $P_1$ and $P_2$ respectively. One of each of the hadron's constituents is taken with momentum fraction $x$. Figure inspired by that from [2].

process into several stages and attempt to efficiently calculate the possible outcomes at each stage. We will briefly detail these below and a graphical representation of these processes is given in Figure 3.1.

**Hard process:** Event generators generally begin with the hard scattering processes, including corrections in perturbation theory, which define the main collision process and immediate outcomes. At hadronic collisions, two incoming particles are selected with a given momentum fraction of the hardon and the matrix element for a given collision process is calculated. The calculation of matrix elements was covered in Section 2.2. At tree-level, event generators will often have in-built matrix element generators (such as AMEGIC [52] and COMIX [53] in Sherpa). For the more specialised calculation of loop-level terms, they will usually interface with external tools, and make calls to these tools when calculating observables — the number of calls is determined by the complexity of the functions, the desired uncertainty on the integration, and the method of sampling the initial and final state particle phase-space (see Section 3.3). Tools for calculating virtual corrections include those which perform integral evaluations [54–56], integrand reduction [54,57–60], those which act as a library of amplitudes [61–65], or generate the amplitudes automatically [66–70]. [1] We discuss matrix element calculation in more detail in Chapters 5 and 6.

**Phase-space integration:** To calculate the cross-sections of a given process, the hard scattering process and it's constituent matrix elements must be recalculated at different phase-space points. This is the fundamental idea behind Monte Carlo integration — the integrand is evaluated multiple times and the results summed to approximate the integral. Event generators generally have built-in tools for efficiently generating new phase-space points to optimise convergence. Monte Carlo integration techniques will be discussed in more detail in Section 3.3.

**Parton shower:** After the hard scattering process, generators seek to include further corrections through parton showering which describes the evolution of the hard scale down to the hadronisation scale. Here, parton showering algorithms are

[1]This categorisation was taken from Chapter 3 in [25].

employed to iteratively emit QCD and QED particles from the initial and final coloured states in the hard scattering process. The particles which remain after showering are then at relatively low scales. The combination of the hard process with the parton shower is performed through matching algorithms [71, 72], which ensure there is no double counting matrix elements.

**Hadronisation:** Finally, at low scales, event generators carry out the non-perturbative procedure of hadronisation. This stage is particularly important since the confinement property of the strong interactions guarantees that the final state particles in hadron collisions must also be hadrons, and so various methods are used to ‘cluster’ the products of the hard scattering and parton showering processes into colourless hadrons.

In this thesis we will be most concerned with the hard scattering processes — specifically, designing and testing methods to replace current matrix element libraries for complex processes for which call times are significant. As part of this, we will introduce some of the commonly used Monte Carlo integration techniques and how event generators arrive at observable values. However, we will address these techniques in the context of the hard scattering processes and so will not discuss the mechanics of the parton showering and hadronisation processes.

## 3.3 Monte Carlo Integration

In order to evaluate the various quantities relevant to theoretical particle physics, such as the observables specified in Equations 3.1.1 and 3.1.2, multiple integrals must be calculated. The complexity of the integrands, along with their multidimensionality, can make these expressions extremely challenging to integrate analytically. While there is much ongoing work to find analytic expressions for higher order processes, even if these are found, implementing them in existing matrix element libraries becomes increasingly challenging, and the speed-up gained from the analytic result may be negligible in comparison to the numerical equivalent.

To address the challenge of evaluating complex multidimensional integrals with no known analytic solution, we use Monte Carlo integration. This technique will not yield the exact result; however, methods have been developed that can quantify the uncertainty in the numerical integral evaluation: simply, in order to improve the precision of a calculation, more compute time is needed to evaluate the integrand at more points. In this section we will briefly introduce the high level concepts of Monte Carlo integration, and discuss some of the more nuanced techniques to improve it.

### 3.3.1 Basics

The fundamental principle of Monte Carlo integration is that for a given function, $f(x)$, of arbitrary number of dimensions, $x \in \mathbb{R}^d$, the integral, $I$, over the unit hypercube, $[0,1]^d$, can be approximated by evaluating the function repeatedly at randomly selected points, $r_i \in [0,1]^d$

$$I = \int \mathrm{d}x f(x) \rightarrow \frac{1}{N}\sum_{i=1}^{N} f(r_i) = \langle I \rangle \,, \tag{3.3.1}$$

where $N$ is the number of times the function is evaluated, and $\langle I \rangle$ is the numerical approximation of the analytic integral $I$. The law of large numbers therefore ensures that this integral approximation approaches the true value as $N \rightarrow \infty$.

This approach is not just limited to the unit hypercube. Any function $g$, integrated over the space $[a,b]^d$, can be mapped to $f(x)$ integrated over the unit hypercube by transforming the integral and incorporating the corresponding Jacobian expression. While it may not always be trivial to actually perform this mapping, the theoretical ability to do so is powerful as it limits the sampling space of input points to the unit hypercube.

As a measure of the quality of convergence to the analytic result, we define the

Monte Carlo estimate of the variance of the function, $\sigma^2(f)$ [1] [2]

$$\sigma^2 = \frac{1}{N}\left[\frac{1}{N}\sum_{i=1}^{N} f(r_i)^2 - \frac{1}{N}\left(\sum_{i=1}^{N} f(r_i)\right)^2\right] \tag{3.3.2}$$

$$= \frac{\left\langle I^2 \right\rangle - \langle I \rangle^2}{N}. \tag{3.3.3}$$

The *Monte Carlo error* is then defined as the standard deviation, $\sigma$, and is often quoted in particle physics calculations. [3] From Equation 3.3.2, we can clearly see that the Monte Carlo error scales as $1/\sqrt{N}$ under this construction, which provides a good estimate for the number of function evaluations required to reduce the Monte Carlo error to a negligible value in comparison to other uncertainties coming into the calculations (e.g. the theoretical uncertainty from scale choices).

### 3.3.2 Reducing the variance

Now that we have defined a measure by which to assess the performance of our integration technique, we can develop methods to optimise for it. Since the function, $f$, may be costly to evaluate (as in the case for higher order and/or high multiplicity matrix elements) we want to minimise the Monte Carlo error while requiring as few evaluations of the integrand as possible. There are many available techniques for doing this, of which we will only discuss two of the most relevant here.

*Stratified sampling* is a simple, yet potentially powerful, method for reducing the Monte Carlo error and gaining faster convergence to the exact integral value. The idea is to divide the integral phase-space into sub-regions, perform integration in each region, and then sum together the results at the end. If a careful choice in

[1] It is worth noting here that that for a reliable estimate for the error, we require the function, $f$, to be square-integrable.

[2] There is discrepancy in the literature as to whether the factorised denominator in the variance is $N$ or $N-1$. Since we are technically sampling from a population, $N-1$ is the correct choice. However, for simplicity we have chosen $N$ throughout this thesis since they are equivalent in the large $N$ limit.

[3] Note here that we will use $\sigma$ to denote something other than the cross-section. While this is still not ideal, it is standard to use this notation for both the variance (where $\sigma$ itself is the standard deviation), and the cross-section. We hope that the context will make clear which of the two the $\sigma$ refers to.

the way the region is sub-divided is made, substantial speed-ups in convergence can be gained. For example, in particle physics processes, the integrand may contain functions with peak and/or divergent structures such as

$$\frac{1}{s^n} \text{ or } \frac{1}{(s^2 - m^2) + \Gamma^2 m^2}, \tag{3.3.4}$$

which correspond to propagators and Breit-Wigner-type resonances, where $s$ is the invariant mass which is integrated over, $m$ is the mass, and $\Gamma$ the decay width. In these examples, a naive sampling of the phase-space which sparsely samples in regions around the peak will have a high variance. However, careful sub-division of the phase-space into, e.g. a smaller localised region around the peak which is more heavily sampled, and then a non-divergent region which can be more sparsely sampled while still producing a small Monte Carlo error, could mean fewer phase-space points are required.

*Importance sampling* is another commonly used technique to achieve a faster rate of convergence during Monte Carlo integration. This method relies on introducing another function, $g(x)$, which can approximate the complex structures in the integrand. The integral to be evaluated then becomes

$$\begin{aligned} I &= \int \mathrm{d}x f(x) \qquad (3.3.5)\\ &= \int \mathrm{d}x \frac{f(x)}{g(x)} g(x) \qquad (3.3.6)\\ &= \int \mathrm{d}G(x) \frac{f(x)}{g(x)}, \qquad (3.3.7) \end{aligned}$$

where

$$g(x) = \frac{\partial^d}{\partial x_1 ... \partial x_d} G(x). \tag{3.3.8}$$

If $g(x)$ is normalised, and we ensure it is positive everywhere, then we can interpret it as a probability distribution. A phase-space sampler can then be built to distribute points according to this distribution which will produce an estimate of the integral

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(r_i)}{g(r_i)}, \tag{3.3.9}$$

where now the points $r_i$ are distributed according to the probability distribution $g(x)$. The Monte Carlo estimate of the variance now becomes

$$\sigma^2\left(\frac{f}{g}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(r_i)}{g(r_i)}\right)^2 - \langle I\rangle^2 \,. \tag{3.3.10}$$

From Equation 3.3.10 we can see that the best way to reduce this error is to choose a function, $g(x)$, such that it best approximates $f(x)$. In practice, we leverage our knowledge of particle physics processes, and known analytic expressions (such as those in Equation 3.3.4), to construct functions which approximate the broad characteristics of $f(x)$, and use these in conjunction with more complex processes such as *adaptive sampling* methods and *multi-channeling* (see Section 3.4).

## 3.4 Integration in Practice

So far we have discussed some of the fundamentals of Monte Carlo integration and the basic underlying techniques. These have been introduced at a theoretical level, but not deeply discussed as practical implementations. In the context of particle physics, we generally want to integrate over some phase-space of initial and final state 4-momenta to calculate observables such as those in Equations 3.1.1 and 3.1.2. This phase-space is constrained by known physics, such as momentum conservation, which limits the allowed free parameters over which to integrate. As mentioned above, we have so far discussed techniques to improve Monte Carlo integration, including stratified and importance sampling. However, in practice we may not have the required advanced knowledge of the integrand behaviour to make use of these techniques out-of-the-box.

One of the simplest implementations of a phase-space integrator often used in particle physics is the RAMBO algorithm [6], which distributes points uniformly and isotropically. Another common technique is the VEGAS algorithm [73, 74], which is used to optimise integration processes and brings together many of the approaches from importance and stratified sampling discussed above. These two approaches will be used later in this thesis and are discussed in more detail in Appendix B.

In reality, these techniques are used alongside other, more advanced, practices to ensure rapid and stable integration convergence. For example, VEGAS can be used with other techniques such as multi-channeling [75], which uses known physics knowledge of the integrand peak structure to help guide adaptive integration methods. In addition, integrators have been designed for specific processes, such as the HAAG [76] and SARGE [77] algorithms, which utilise knowledge of pure QCD processes to achieve faster integration. These more advanced and specific techniques will not be the focus of this thesis.

# Chapter 4

# Machine Learning for Event Generation

The field of machine learning is vast, with many diverse theoretical constructions and applications. ML techniques have found many applications in particle physics, with one of the first uses being fitting PDFs by the NNPDF collaboration [78]. In this Chapter, we will introduce the key concepts of machine learning in the field of particle physics, with a particular focus on applications to two of the largest bottlenecks in theoretical particle physics calculations: phase-space sampling, and matrix element calculations. Gaining efficiency in these two areas could significantly assist in calculations involving higher order terms in QCD, and their inclusion in full event generation. Indeed, ML techniques to address end-to-end event generation itself will also be briefly covered in this Chapter. Restricting the discussion to these applications in particle physics will self-select which ML methods will be introduced. Specifically, we will focus on decision trees, including random forests and boosting/bagging methods, and neural networks and their uses within generative techniques. For a living review of the uses of ML in particle physics more broadly see [79].

In this thesis, we shall concentrate on uses of machine learning for regression problems. Specifically, let $f$ be some machine learning model for regression tasks which takes input data, $x$, in the form of a $d-$dimensional vector and returns a single real number, $y^*$, then

$$f(x) : \mathbb{R}^d \to \mathbb{R}, \;\; x \in \mathbb{R}^d \mapsto y^* \in \mathbb{R}. \tag{4.0.1}$$

We will also only discuss these algorithms in the context of supervised learning, i.e. where models learn how to provide a good output for a given input by being trained on pairs of inputs and 'correct' outputs — *training data.* Throughout this Section, we will assume the training dataset, $\mathcal{D}$, is comprised of $n$ pairs of input vectors, $x$, and outputs, $y$, such that

$$\mathcal{D} = \{(x_i, y_i) \,|\, x \in \mathbb{R}^d, y \in \mathbb{R}\}, \;\; N = |\mathcal{D}|. \tag{4.0.2}$$

In its simplest form, ML is an optimisation problem in which the learning procedure is defined through the minimisation of a loss function, $L$, by modifying tunable parameters of the model. The construction of the loss function and the optimisation procedure can be model and application specific, examples of which will be described briefly in this Chapter. Machine learning models can also have many hyperparameters which can be tuned to further optimise performance. The training data is generally used to optimise the model for a specific set of hyperparameters, and *validation data* is used to optimise the hyperparameters. Finally, *testing data* is used to assess model performance. This is generated independently of the training and validation datasets and is not used for any optimisation. This Chapter is based on and has been inspired by multiple sources [3, 80].

## 4.1 Decision Trees

Decision tree algorithms [81] take a vector input of categorical or continuous variables and return a 'decision'. The decision tree algorithm breaks down the input space into subsets while iteratively constructing the tree. The tree is made up of decision nodes and leaf nodes. Each node is associated with a region of input space, with decision nodes controlling the splitting of the input space into sub-regions, while

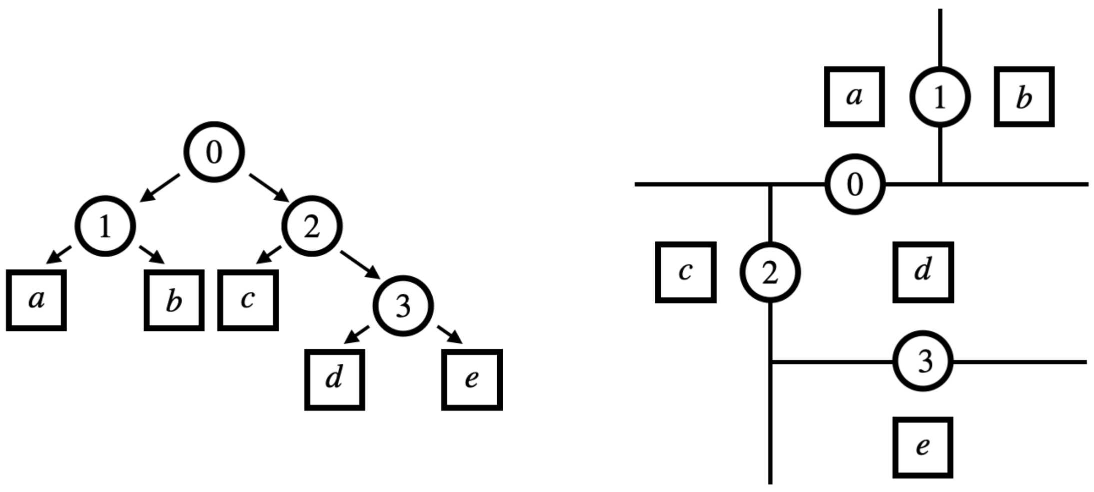


Figure 4.1: Right: a decision tree construction with decision nodes represented by circles, and leaf nodes represented by squares. Right: the decision nodes split the input space into sub-regions in which the leaf nodes make the final decision. Inspired by Figure 5.7 in [3].

the leaf nodes are where the final decisions are made. In the case of regression, each leaf node has a linear function of some subset of the input variables, and the learning algorithm must decide how many times to split the input space and apply linear regression. Figure 4.1 gives a graphical representation of such a decision tree construction. In the example of approximating the matrix element for a specific process, given an input vector of phase-space 4-momenta, the decision tree could be thought of as dividing the phase-space and approximating sub-regions with linear functions of the input variables.

As with all functional approximations, there are various ways in which decision trees can be optimised. A common way to do this is to maximise the *information gain* at each decision node to define the split. How much information is gained about $Y$ (for example, the value of the target variable) by knowing something about $X$ (a value of a certain input feature for a given data point) is defined as

$$IG(Y,X) = H(Y) - H(Y|X), \tag{4.1.1}$$

$$H(X) = -\sum_{x\in X} p(x)\log_2(p(x)), \tag{4.1.2}$$

(4.1.3)

where $H$ is the entropy, and $p(x)$ is the probability of finding an element of the training dataset in one of the sub-regions, $x$, after splitting. By calculating the information gain over all possible splittings at a decision node, the spitting maximising the information gain is chosen for that node.

Such procedures provide a possible decision tree which can be used for *inference*. However, an individual tree is considered a *weak learner* — it is relatively naive in its setup, has the propensity to overfit to the training data, and may not perform much better than random guessing. To address this, various methods have been developed, many of which leverage the idea of model *ensembling*. A survey of decision tree learning methods can be found in [82].

### 4.1.1 Bagging and Random Forests

The simplest ensembling technique is to train multiple models and average their results — this is the fundamental idea of *bagging* (otherwise known as Bootstrap Aggregation). When performing bagging, subsets of the training data are randomly sampled and a different decision tree trained on each subset. During inference, the results from each tree are averaged to give a final numerical approximation. Training each tree on a subset of the data and averaging the results reduces the chance of overfitting to the training data since no one tree 'sees' the whole dataset.

*Random forests* use bagging with a modification [83]. Instead of just having trees trained on different subsets of the training data, each tree also only uses a subset of the input features. This technique allows for better handling of high dimensional data as well as missing values (which can be common in large datasets). The downside of both random forests, and bagging in general, is that the process of averaging the results means loosing precision in the output of the model, and the training phase takes longer than training a single model.

### 4.1.2 Boosting

Bagging trains trees in parallel and averages their results. However, *boosting* trains trees sequentially with the result of one *strong learner* — a model of arbitrarily good performance depending on the training regime. Each tree in the sequence learns using the input data weighted by the error of the previous tree (or sequence of trees) — i.e. if the previous tree (or sequence of trees) is a good approximation of one subset of the training data, but is a poor approximation of another, the next tree in the sequence will be trained on data more heavily weighted towards the poorly approximating data.

The training of subsequent trees in the sequence can be improved through various optimisation algorithms, the most common of which is *gradient boosting* [84,85] where a gradient descent algorithm can optimise for a given differentiable loss function (see Section 4.2) to guide the construction of future trees. As an example of the use of boosted decision trees in the context of event generation, methods have been developed for gradient boosted regression integration where a series of decision trees is used to approximate the probability distribution, $g(x)$, defined in Equation 3.3.6 [86]. This method was shown to allow for the integration of functions for which traditional algorithms have previous failed. Similar methods for matrix element approximation have been developed using the parallelised boosted decision tree approach, XGBoost [87]. This methodology was tested on the loop-induced $gg \to ZZ$ process at leading order, demonstrating the potential for large speed-ups in matrix element calculations [88]. Given that decision trees have the potential to output 'human-readable' explanations for their decisions at each node, it would be interesting to explore these to analyse how the phase-space is divided by a decision tree in comparison with similar methods such as VEGAS (see Appendix B.2).

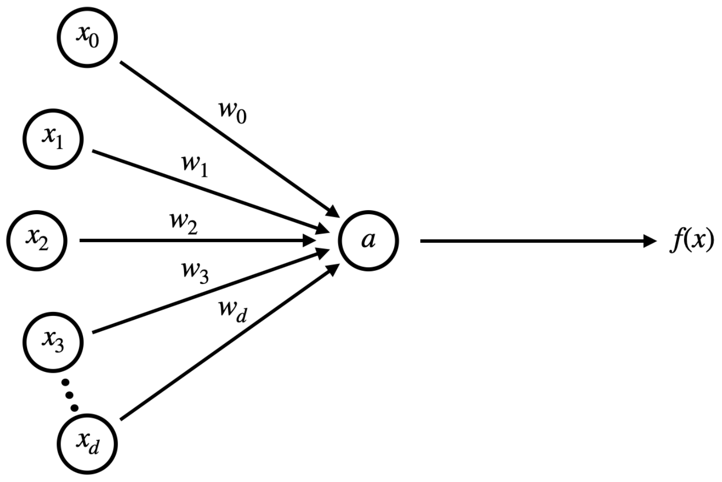


Figure 4.2: Construction of the perceptron. Input variables, along with a bias term, $x_0 = 1$, are weighted, combined together, and pass through an activation function, $a$, which results in the output, $f(x)$.

## 4.2 Neural Networks

Neural networks (NNs) are a commonly used methodology due to their ability to approximate highly non-linear functions. There are many applications and constructions for NNs — in this introduction we shall focus on Fully Connected Networks (FCNs) which comprise a series of layers containing *perceptrons*. In this Section we shall introduce these concepts and discuss their optimisation.

### 4.2.1 The Perceptron

The perceptron [89] is the simplest building block for neural networks. Given an input data point, $x = (x_1, ..., x_d)$, the output of the perceptron is given by

$$f(x) = a\left(\sum_{k=0}^{d} x_k \cdot w_k\right), \tag{4.2.1}$$

where $w_k$ are the weights which control the relative importance of the input variables, and $a$ is the activation function. Activation functions can take on a range of values and the choice of function is usually based on the data types and problem to be addressed. For now, we will assume that $a$ takes the form of a linear function, which is the most common choice for activation functions just before the output is given in regression problems. The sum in Equation 4.2.1 runs from zero because we introduce a fixed variable to each input's data points, $x_0 = 1$. This zeroth element means the term $x_0 \cdot w_0 = w_0$ serves as a bias constant. Figure 4.2 shows a schematic of this representation.

In practice, we optimise the weights such that the function is a good approximation of the training data. As a result of the increased efficiency in automated differentiation libraries, *gradient descent* algorithms have become the norm in machine learning for optimisation problems. [1] We therefore define a loss function, $L$, which provides a metric to be optimised. In regression tasks, a common choice is the mean squared error (MSE)

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2, \tag{4.2.2}$$

The gradient of the loss function is then calculated to update the weights

$$w(t+1) = w(t) - \eta \nabla_w L, \tag{4.2.3}$$

where the weights, $w = (w_0, ..., w_d)$, at time $t+1$ are updated from their values at time $t$ by taking the gradient of the loss function with respect to the individual weights and multiplying this by the *learning rate*, $\eta$. The learning rate plays an important role in gradient descent algorithms and controls the importance of the gradient. In this implementation, the learning rate is considered a hyperparameter which can be separately tuned. However, as the network approaches the global minimum of the loss landscape during optimisation, setting this parameter too large may cause the network to overshoot this value. Algorithms for adapting the learning rate dynamically during training can be used to address this problem. Common choices include: AdaGrad [91], which scales the model parameters in proportion to the historical values of the gradient squared, thereby giving parameters with larger gradients large learning rates and vice versa; RMSProp [92] adapts the AdaGrad algorithm to put less weight on historical gradients in the distant past; and Adam

[1]For an overview of gradient descent algorithms commonly used in the context of ML, see [90].

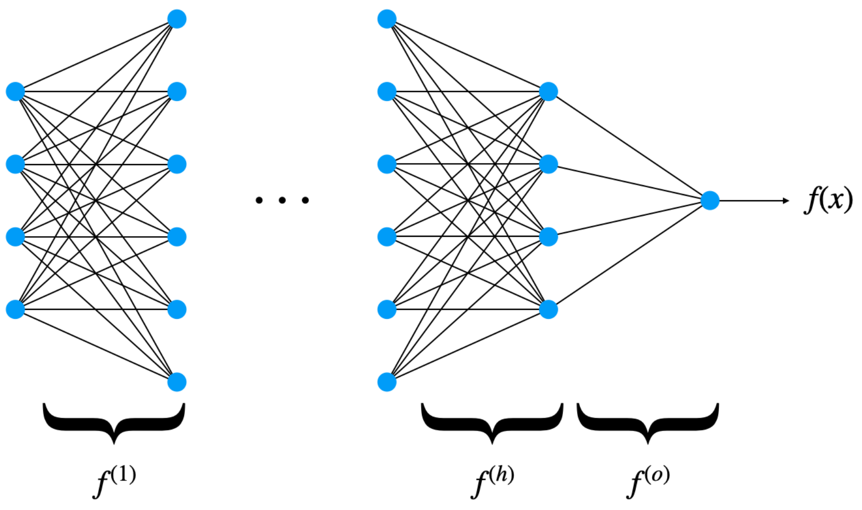


Figure 4.3: Construction of a typical Fully Connected Network for regression with a single number output. Input variables, along with a bias term, are weighted and combined together at each node in the next layer. All nodes in each layer are connected to all other nodes in subsequent layers. This continues for all hidden layers, $\{f^{(1)}, ..., f^{(h)}\}$ and terminates once we reach the output layer $f^{(o)}$. Each node in each layer can have a different activation function associated with it. It is common practice to assign the same activation function to all nodes in the hidden layers, with the final activation function usually chosen to be linear for regression problems.

optimisation [93], which is similar to RMSProp with the addition of *momentum* [94] that adds an additional correction to the gradients based on previous values.

From this description it is clear that the perceptron is a simple object which can only fit functions comparable to the chosen activation function. However, by combining multiple perceptrons together in parallel and in series, with interlinking weight *layers*, we can access much higher dimensional functions with complex non-linearities.

## 4.2.2 Fully Connected Networks

FCNs are composed of multiple perceptron-like objects arranged in layers — layers in between the input and output layers are referred to as *hidden layers*. Activation functions introduce non-linearity into the network and common choices in regression problems are hyperbolic-tangent, sigmoid and rectified linear unit (ReLU) functions

$$a(x) = \tanh(x), \;\; a(x) = \frac{1}{1+e^{-x}}, \;\; a(x) = \max(0, x).$$

Different functions have been found to have different advantages and use cases. For example, the sigmoid function may be useful if the desired output is bounded between $(0, 1)$, whereas the ReLU function has been found to ensure rapid convergence in classification tasks [95]. The output of a network with $h$ hidden layers is then given by

$$f(x) = f^{(o)} f^{(h)}(f^{(h-1)}(...(f^{(1)}(x)))), \tag{4.2.4}$$

where $f^k$ are the outputs of layer $k$, and $f^o$ is the output layer which is usually given a linear activation function in the case of regression. A schematic of this is given in Figure 4.3. The choice of activation function is dependent on the problem at hand and can be optimised during the hyperparameter optimisation phase of architecture development (along with other hyperparameters such as the number of hidden layers and the number of nodes in each layer). Since the activation functions interact in a non-trivial way, simplifying their choice by assigning them to be the same for all nodes in the hidden layers helps minimise undue complexity, and drastically reduces the number of possible hyperparameter combinations.

Optimising for a given loss function uses the same logic as applied in Section 4.2.1 — by using back-propagation, which is a simple application of the chain-rule to calculate the Jacobian with respect to each weight in the network. Indeed, by leveraging matrix linear algebra, we can simply write the weight update as

$$W(t+1) = W(t) - \eta \nabla_W L, \tag{4.2.5}$$

where the weights are now represented as a matrix $W$ [96].

From Equation 4.2.4 it is clear that the layering of multiple hidden layers containing a number of nodes with different non-linear activation functions allows such models to approximate highly complex functions. Furthermore, advances in computational matrix algebra and automated differentiation, along with hardware advances such as in graphical processing unit (GPU) technology, mean neural network training times are continuing to be drastically reduced. [1] [2] This is partly due to the large parallelised arithmetic which can be naturally performed on GPUs, as well as the building of specialised ML software optimised for GPUs and the production of GPUs specifically designed for the high dimensional matrix algebra performed frequently in ML tasks. It is worth noting, however, that much attention has recently been on speeding up highly time and memory consuming computer vision tasks, whereas the comparable order(s) of magnitude speed gain from using many GPUs is not yet widely observed when training models for numerical regression.

Given this ability to approximate complex functions, neural networks have been applied to phase-space sampling and integration [98,99], where they have been shown to increase the speed at which functions can be integrated. Similar to applications of boosted decision trees to this problem, in these examples neural networks are trained to map a random number to the probability distribution, $g(x) \in [0,1]^d$, for use in importance sampling. However, since the network acts as a variable transformation, at each point we must calculate its gradient to determine the Jacobian, which can become computationally expensive. To address this, recent work [100–102] has attempted to use normalising flows [103, 104]. These have been shown to avoid the computational cost of calculating the gradient of the network, when determining the Jacobian through the use of coupling layers [104, 105]. However, the performance of these approaches has been shown to decrease as the multiplicity of the process increases. Recent work using auto regressive flows [106–108] (which are related to their normalising counterparts in approach but are expected to perform better on larger feature spaces) applied to phase-space sampling [109]. Here, the authors test their approach on $e^+e^- \to t\bar{t}$ at LO and $pp \to t\bar{t}$ with parton showering at NLO. Promising results were found when trained on weighted events, with too

[1]For a survey of automatic differentiation approaches in the context of machine learning see [97].

[2]We will show assessment of the speed of classical analytic and numerical function evaluations compared with neural network inference in the context of matrix element calculations in Chapters 5 and 6.

many point being rejected during unweighting for good performance on these events. The incorporation of negative weights was also tested, with the model found to incorporate these and produce results largely within the Monte Carlo error of the 'true' distribution.

NN based approaches have also been developed to address other components of Monte Carlo event generators including parton showering [110–112] and event reweighting [113]. Similarly, several works have focused on developing NN techniques for explicitly learning the cross-section of specific processes [114, 115].

## 4.3 Generative Networks

Generative Adversarial Networks (GANs) [116] consist of two NNs — a *generator* and a *discriminator* — and are based on a game theoretic scenario in which these two networks compete. The exact details of this methodology are beyond the scope of this thesis. However, we give a brief overview of how GANs can be constructed and their applications to event generator simulations. For more details see the original paper [116] and Chapter 20 of [3].

The aim of the generator, $g$, is to produce samples $x^* = g(z)$, given some input 'noise', $z$, which is usually drawn from a Gaussian distribution. The discriminator, $d$, is either fed the output of the generator, or a sample from the training dataset, $x$, and attempts to determine from which source the input sample originated. [1] By training these two networks to compete against each other, a generator can be obtained which is able to produce samples that are hard to distinguish from reality (the training data).

The simplest way to train these networks is as a zero-sum game. Let's assume the function $v(g, d)$ determines the payoff of the discriminator and $-v(g, d)$ the payoff

[1] In applications of GANs relevant to this thesis, the generators perform a regression task, whereas the discriminator is a classifier. As mentioned at the beginning of the Chapter, we will not discuss the details of ML classification. However, in the case of GANs, it is sufficient to think of the output of the discriminator as being a single probability score which estimates the network's belief that the sample is synthetic or from the training dataset.

of the generator. Each network then seeks to maximise its payoff such that at convergence

$$g^* = \mathrm{argmin}_g \mathrm{max}_d \, v(g,d), \tag{4.3.1}$$

where a common choice is

$$v(g,d) = \mathbb{E}_{x\sim p_{\mathrm{x}}}[\log(d(x))] + \mathbb{E}_{z\sim p_z}[\log(1-d(g(z)))], \tag{4.3.2}$$

where $p_x$ is the training data distribution and $p_z$ is the generator input noise distribution [116]. Writing this in the same format as Equation 4.2.2, the loss functions to be minimised are

$$L_d = -\frac{1}{N}\sum_{i=1}^{N}\left[\log(d(x_i)) + \log(1-d(g(z_i)))\right], \tag{4.3.3}$$

$$L_g = \frac{1}{N}\sum_{i=1}^{N}\left[\log(d(x_i)) + \log(1-d(g(z_i)))\right], \tag{4.3.4}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\log(1-d(g(z_i)))\right], \tag{4.3.5}$$

where $L_d$ and $L_g$ are the discriminator and generator loss functions respectively and the final lines comes from the fact that the generator can not directly affect the $\log(d(x_i))$ term.

In practice, GANs which use NNs suffer from the non-convexity of $\mathrm{max}_d \, v(g,d)$, which can make training noisy and convergence challenging. Other formulations of the training procedure continue to be developed in an attempt to address the problem (see [117, 118] for examples); however, these are beyond the scope of this thesis to discuss in detail. Despite these challenges, GANs have important applications in multiple fields, including various components of event generator simulations.

Full event generation can be highly time consuming, and so GANs have been developed to replace the full pipeline described in Chapter 3. Many works have taken off-the-shelf GANs with FCN architectures to emulate full event generation, including detector simulation, for a variety of processes [119–123]. However, in [124] the authors use a convolutional neural network (CNN) [125, 126] architecture in place of the FCN setup. [1] Physics inspired modifications of traditional loss functions have also been developed to help GANs learn relevant features. In [127] the authors directly include the masses of the $Z$-boson to aid peak reconstruction which results in an analysis-specific loss function, while others use knowledge of the peak structure of integrand for a more generalisable approach [128]. In the latter example, the authors do not include detector simulations in order to more fully probe the ability of the networks to learn complex intermediate resonances present in $pp \to t\bar{t} \to (bq\bar{q}')(\bar{b}\bar{q}q')$. The same loss function was used to study electron-proton collisions without detector simulations in [129]. In this work the authors also developed feature transformations, while also augmenting the input to the discriminator with additional features, to improve their performance. It would be beneficial to rigorously test the generalisability of these physics informed additions and alterations which have been found to boost performance.

In many of the above examples of ML applications to particle physics processes, it is reasonable to ask to what extent the models can encode physics beyond that contained in the training data. In the case of GANs this questions was addressed in [130], where the authors applied GANs to a range of toy models representative of physics processes. The authors trained the models on a small training dataset drawn from a known probability distribution and then generated data points using the trained GAN until the information added by the network saturated. The authors found that the GANs were indeed able to "amplify" the training data to a certain extent before the addition of more generated data points no longer helped the generated distribution approximate the known distribution. This has important ramifications for the use of ML models in particle physics processes, although more testing must be done to assess this on realistic distributions, as well as the effects of training dataset size on the amplification achievable.

In addition to full event generation, GANs have been applied to event unweighting

[1] CNNs can be thought of as a natural extension of the neural networks discussed in Section 4.2, where each input is a portion of an image which is then convolved with a weight matrix (or kernel). For an introduction to CNNs see Chapter 9 of [3]

[131] and subtraction [132], with early work assessing its performance for phase-space integration of toy examples where traditional methods have struggled [86]. Recent works have discussed incorporating Bayesian methods for uncertainty estimation — a key ingredient for physics use cases of these methods which will also be discussed in more detail (albeit from a frequentist perspective) in Chapter 5 — into these generative methods [133] building on previous work which used similar techniques for the extraction of energy of a tagged top quark inside a fat jet [134]. For a more in depth review of some of the works discussed in this Chapter see [135, 136], and see [137] for a discussion on how GANs can be used more broadly for event generation and fast simulations in High Energy Physics (HEP).

It is worth noting that Variational Auto-Encoders (VAEs) [138] can also be used in a similar way to GANs and many of the works discussed above have testing VAE methods in parallel. More information on VAEs more broadly can be found in [138, 139].

# Chapter 5

# Machine learning for matrix element approximation: $e^+e^- \to q\bar{q} + \text{jets}$

## 5.1 Motivation

Phenomenological studies of high multiplicity final states at collider experiments present a substantial theoretical challenge and are increasingly important ingredients in experimental measurements. During the last 15 years, a dramatic improvement in computational algorithms for one-loop amplitudes has led to a number of highly automated codes capable of predictions at NLO accuracy in the SM [51, 61, 70, 140, 141].

These codes are based around numerical algorithms that bypass the growth in algebraic complexity that analytic approaches suffer from. As discussed in Chapter 1, the computational cost of these algorithms is still relatively high, resulting in huge commitment of CPU and personnel resources to obtain the necessary theoretical predictions for current experiments.

In this Chapter, we begin to explore one way in which we can use ML technology,

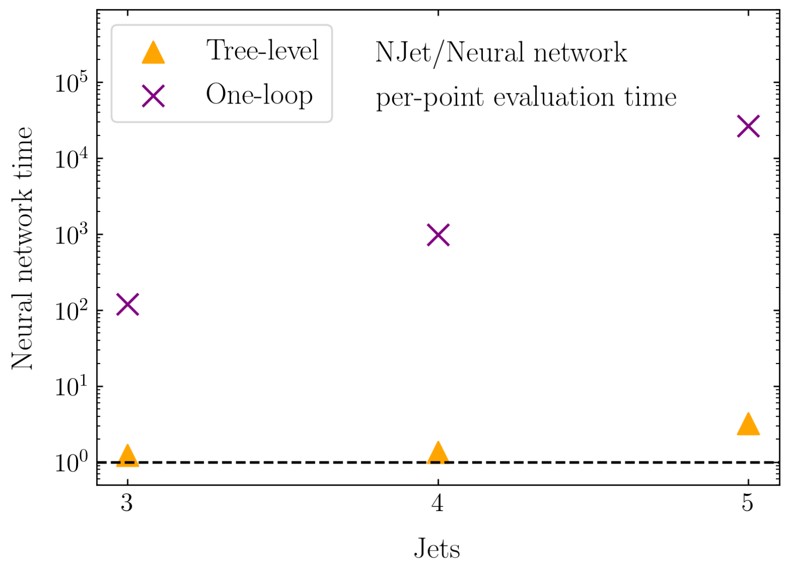


Figure 5.1: A ratio of the CPU cost to calculate tree-level and one-loop amplitudes in NJet to inferring on a neural network (built in Keras/TensorFlow) as a function of the number of legs (equivalently number of variables). The black line denotes 1. This demonstrates the fairly trivial fact that the neural network is fast to call compared to numerical equivalents.

in the form of neural networks (see Section 4.2), to decrease the computational cost of precision simulations. In particular, we consider high multiplicity scattering processes, with high mathematical complexity, where it is less clear how to make use of conventional interpolation methods such as polynomial fits and interpolation grids [142–146].

Neural networks have the potential to provide extremely fast and lightweight approximations of complicated amplitudes. In Figure 5.1 we demonstrate this for the particular test cases which are the subject of this Chapter — the tree-level and one-loop amplitudes inside the NJet amplitude generator [65] for $e^+e^- \to \leq 5$ jets. [1] Here, we see that the neural network is fast to call and has a very mild dependence on the number of variables. The challenge is to train the network well enough that it can be interpolated and extrapolated reliably over a complete range of differential observables. While the potential speed up in the function call is quite striking,

[1]An example of a matrix element calculation for this process is given in Section 2.2.

the real challenge is not clear from this analysis. The actual improvement in CPU cost must include the time takes to train the network such that interpolation and extrapolation are sufficiently accurate and reliable.

In this Chapter we design a deep learning pipeline to approximate $e^+e^- \to \leq 5$ jet matrix elements at both LO and NLO, thus exploring processes with significantly higher multiplicity than those considered previously (see Chapter 4 for a discussion of prior work). While [114] uses a more automated approach for phase-space sampling to aid in training a neural network, we employ physics-based knowledge of the processes in designing our pipeline. We analyse the effectiveness of this approach and what this might tell us about the phenomenological set up. We pay careful attention to the errors and uncertainties in our neural network approximation, and offer a comprehensive implementation of neural network regression analysis.

The techniques developed in the Chapter will be applied to the $e^+e^- \to \leq 5$ jet processes, and will be further developed for more complex loop-induced processes in Chapter 6.

## 5.2 Computational setup

We use NJET [65] (an on-shell based C++ code) to evaluate colour and helicity summed Born and virtual matrix elements for $e^+e^- \to \leq 5$ jets, denoted $\mathcal{M}^{(n,0)}$ and $\mathcal{M}^{(n,1)}$ respectively. Going beyond the simple analytic approaches presented in Chapter 2, NJET uses integrand level reduction [147] and generalised unitarity [61, 148–153] to construct loop amplitudes from tree-level input, which is computed efficiently with Berends-Giele recursion [154]. For a given phase-space point, NJET calculates the virtual and Born matrix elements, along with the $1/\epsilon$ and $1/\epsilon^2$ correction coefficients (arising from dimensional regularisation discussed in Chapter 2). In this thesis we do not use the correction coefficients and instead only focus on the matrix elements from which we can calculate the k-factors:

$$\text{k-factor} = \frac{|\mathcal{M}^{(n,1)}|^2}{|\mathcal{M}^{(n,0)}|^2}. \tag{5.2.1}$$

It should be noted that the definition of the k-factor in Equation 5.2.1 is not the conventional ratio of the full NLO to LO calculation, but rather the ratio of the matrix elements. In addition, our discussion of NLO matrix elements will be limited to the virtual corrections only, and not the real radiation part.

For ease of use, NJET is interfaced via the Binoth Les Houches Accord (BLHA) [155,156]. The BLHA is designed to provide a standardised interface between Monte Carlo tools and matrix element programs. We leave the implementation of interfacing with event generators to Chapter 6.

We explore the performance of various neural network generated amplitudes for total and differential cross-section computations at LO, as well as their respective k-factors at NLO. We find that as the multiplicity increases, IR singularities on the edge of the phase-space increasingly cause problems for a single neural network, which struggles to find a good fit across the whole phase-space. To improve the approximation, we divide up the phase-space into sectors according to the FKS subtraction method [37,38]. Although we do not actually perform subtraction, this phase-space decomposition isolates the IR singularities and allows the training of networks to focus on improving performance on each partition individually.

### 5.2.1 Phase-space partitioning for final state singularities

We explore two pipeline configurations: i) we naively train a single network over all sampled points in phase-space; ii) we divide the phase-space into divergent and non-divergent regions in an attempt to partially isolate the IR singularities and then further sub-divide the divergent region according to the FKS subtraction method, training one network on the non-divergent region, and a different network on each partition. For clarity, we will generally refer to the naive single network and partitioned ensemble of networks as 'models', and the individual networks comprising these models as 'networks'.

We parameterise our phase-space according to the Lorentz invariant $y_{ij} = s_{ij}/s_{\text{com}}$, where $s_{ij} = (p_i + p_j)^2$, and define all cuts with respect to this quantity. The partition dividing divergent and non-divergent regions is defined to be at $y_p$. To introduce the concepts of the phase-space partitioning method, we will use a global kinematic cut parameterised by $y_{\text{cut}}$, and then progress to more complex cut configurations which are more relevant for the phenomenological analyses in Chapter 6.

Using these two scales, the divergent region, $\mathcal{R}_{\text{div}}$, and the non-divergent region, $\mathcal{R}_{\text{non-div}}$, are defined as follows:

$$\mathcal{R}_{\text{div}} = \{p \,|\, y_{\text{cut}} \leq \min(y_{ij}) \leq y_{\text{cut}} + y_p,\, p = (p_a, p_b, p_1, ..., p_n),\, i, j \in \{1, ..., n\}\}, \tag{5.2.2}$$

$$\mathcal{R}_{\text{non-div}} = \{p \,|\, y_{\text{cut}} + y_p \leq \min(y_{ij}),\, p = (p_a, p_b, p_1, ..., p_n),\, i, j \in \{1, ..., n\}\}, \tag{5.2.3}$$

where $p$ is a phase-space point consisting of the initial state 4-momenta, $p_a$ and $p_b$, and the outgoing momenta, $\{p_1, p_2, ..., p_n\}$, where $n$ is the number of jets. Given the lack of initial state singularities in $e^+e^-$ collisions, this is an appropriate choice. However, we will generalise this definition to hadronic collisions in Section 6.3.1.

In the FKS subtraction formalism, the phase-space is divided such that the kinematic regions resulting from each partition contain only a specific subset of singularities. In order to achieve this, a set of ordered pairs, known as FKS pairs, are introduced. In our case of $e^+e^- \to\leq 5$ jets we define these as:

$$\begin{aligned}\mathcal{P}_{\text{FKS}} = \{(i,j) \,|\, 1 \leq\ & i \leq n_g + 2,\ 3 \leq\ j \leq n_g + 2, i \neq j, \\ & \mathcal{M}^{(n,0)} \text{ or } \mathcal{M}^{(n,1)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i || \vec{p}_j\},\end{aligned} \tag{5.2.4}$$

where $n_g$ is the number of gluons in the process.

We then construct a partition function similar to that of [157, 158] (for a brief introduction to different FKS pair definitions and partition choices see Appendix C):

$$\mathcal{S}_{i,j} = \frac{1}{D_1 s_{ij}}, \quad D_1 = \sum_{i,j \in \mathcal{P}_{\text{FKS}}} \frac{1}{s_{ij}}, \tag{5.2.5}$$

such that:

$$\mathrm{d}\sigma^{(X)} = \sum_{i,j} S_{i,j}\, \mathrm{d}\sigma^{(X)}, \tag{5.2.6}$$

where, in this example, $\sigma^{(X)}$ represents either the Born cross-section, $\sigma^{(B)}$, the virtual correction, $\sigma^{(V)}$, or the k-factor, $\sigma^{(K)}$.

To demonstrate this partitioning effect, we analyse the process $e^+e^- \to q\bar{q}g$. Here, we can isolate each of the two FKS pairs $\{qg, \bar{q}g\}$ and weight all the phase-space points in the divergent regions according to the behaviour of $\mathcal{S}_{i,j}$ for each pair. The first pair, in principle, corresponds to either the quark and gluon going collinear or the quark or gluon going soft. Since we cannot have soft quarks, this FKS partition only contains the singularities for the soft gluon and collinear quark and gluon. The behaviour of the FKS partition function, $S_{q,g}$ can be clearly seen in Figure 5.2, where we observe increasingly highly weighted points as $s_{qg}$ approaches 0.

An advantage of this method is that the interpolation between singular regions is smooth since they add together to produce the overall cross-section (see Equation 5.2.6). [1] By weighting the matrix elements in this way, phase-space points closer to the $q||g$ singularity contribute with increasing significance to the corresponding neural network's loss during training. A similar analysis can be performed for the second FKS pair in this process.

Since the FKS pairs are ordered, the upper bound on the number of pairs for our processes is

$$N_{\text{max}} = \frac{n(n-1)}{2} - 1, \tag{5.2.7}$$

where $n$ is the number of jets and the $-1$ comes from the fact that $\{q\bar{q}\}$ is not an FKS pair by definition. It should be noted that the number of pairs can be reduced

[1]An alternative implementation would be to partition the phase-space in a piecewise manner according to Heaviside step functions (as in [37]); however, this introduces an additional set of scale choices and significantly reduces the number of phase-space points left for each network to learn the complicated divergent structure. Indeed, we found that when partitioning piecewise the network performs significantly worse in comparison to this smooth implementation.

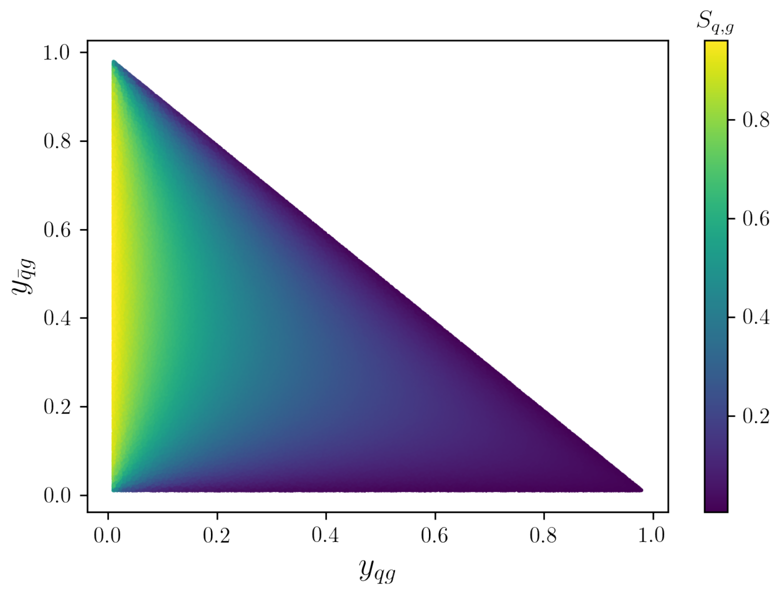


Figure 5.2: Behaviour of the $S_{q,g}$ FKS partition function relative to $y_{ij} = s_{ij}/s_{\text{com}}$.

in reality due to the symmetric behaviour of all gluon-gluon, or quark-gluon pairs; however, for simplicity we partition into $N_{\text{max}}$ regions. For example, in the case of $e^+e^- \to q\bar{q}g$, $N_{\text{max}} = 2$, but since the behaviours of the two pairs in this process are identical, we could reduce this to one.

After using the FKS partition function to divide the region $\mathcal{R}_{\text{div}}$, we are left with $N_{\text{max}}+1$ regions in total across which we train the same number of networks. We find that setting the scale to $y_p = 0.01$ is generally applicable to all processes analysed in this Chapter.

### 5.2.2 Neural network setup

We compare the performance of two neural network setups: firstly, a singular network is trained over the entire uniformly sampled phase-space; secondly, an ensemble of $N_{\text{max}} + 1$ networks is trained over the partitioned phase-space.

### Data

The phase-space is uniformly sampled using the RAMBO algorithm [6], with each point initially having a weighting of unity (see Section B.1). At LO, we train the naive model on data generated from sampling over the entire phase-space uniformly, whereas we train the partitioned model on samples drawn equally from the divergent and non-divergent regions. [1] At NLO, due to the computational expense of virtual matrix element calculation, the phase-space is uniformly sampled as a whole and then divided into $\mathcal{R}_{\text{div}}$ and $\mathcal{R}_{\text{non-div}}$ regions after sampling. RAMBO was chosen for its simplicity, for the ease with which it can be altered to our specifications, and because it highlights interesting pitfalls and difficulties in high-dimensional functional approximations (see more on this below). In total we generate 500k phase-space points for training at LO, but only 100k at NLO due to the complexity of the problem.

The IR poles in the matrix element result in singularities. Neural networks for classification tasks have been repeatedly shown to perform better when datasets are balanced, thus helping to avoid bias in the classification. Balancing can be done through a variety of methods such as over and under sampling, as well as loss function weightings. In regression tasks, the equivalent to class imbalances are under sampled regions that behave significantly differently to the rest of the sampled space. When doing explicit numerical calculations of the matrix elements, these imbalances are not such an issue and their effect when calculating observables can be estimated by the Monte Carlo error and by phase-space resampling; yet they become significant when training a network. Through balancing the training datasets in the divergent and non-divergent regions, and using the FKS partitioning method as outlined above, we hope to address the issue of underrepresented regions. However, this is not as practical to do at one-loop, or during phase-space generation in existing

[1] Testing was done to assess the significance of equally sampling from the divergent and non-divergent regions of phase-space when training the naive model as well, although we found little significant performance increase relative to that of using the partitioned model.

event generators, so this is only performed at LO.

As discussed in Chapter 3, there exist increasingly sophisticated non-machine learning based methods for phase-space sampling which seek to more optimally sample the space for faster convergence. RAMBO, however, is indifferent to these variational differences in phase-space, giving a more naive sampling, yet the ability to construct an interpolation function from a uniformly sampled phase-space means we save computational time during the sampling stage. Although performance of our approximation may be increased using these more sophisticated methods, we demonstrate sufficiently good results while requiring only the use of simple sampling techniques like RAMBO. This further shows the power of our method and the additional time savings it can offer.

Once the phase-space points are generated, we use NJET [65] to calculate the corresponding squared matrix elements at LO, and the virtual correction terms at NLO, for $e^+e^- \to Z^*/\gamma \to q\bar{q} + n_g$. We calculate all quantities in the four-dimensional helicity (FDH) scheme, assuming all external legs to be massless, with the number of light quark flavours set to $n_f = 5$, and use the same renormalisation scale as in [159].

When training the network, the dataset is split in an 80:20 ratio for training and validation. Furthermore, independently generated, unseen datasets are used for testing the performance of our models. Model testing consists of inferring on these unseen test points to create cross-section and differential plots as shown in Section 5.3. Through generating many more points for testing than training we demonstrate the performance of our methodology as an interpolation function by further extrapolating into the divergent region.

To avoid the problem of vanishing/exploding gradients, we standardise our data to zero mean and unit variance at each input node and across the targets. For additional details on methods for preprocessing data, see Appendix D.

### Architecture

Choosing an optimal network architecture is non-trivial due to the large number of parameters that can be tuned to an array of criteria. It is common to approach a singular problem using a neural network and optimise the architecture for that process. However, because we want to demonstrate the ability of networks to become sophisticated multi-parameter interpolation functions, we require these models to generalise to a variety of processes.

For this reason, we do not fine-tune a network to any particular process, but rather attempt to employ the same architecture for each process (in Appendix D we show the results of a hyperparameter scan to verify the choices made here for the case of the diphoton processes discussed in Chapter 6). The neural networks are parameterised using Keras [160] with a Tensorflow [161] backend. They comprise of fully-connected layers with an input layer of $(n-1)\times 4$ nodes and output of 1, with three hidden layers made up of of 20-40-20 nodes. The hidden layers all use hyperbolic-tangent activation functions and the output node has a linear activation function.

The loss function is taken to be the mean squared error,

$$L = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2, \tag{5.2.8}$$

where $N$ is the number of training points, $f : \mathbb{R}^d \to \mathbb{R}$ is the function describing the neural network, $x_i$ is the ith $d$-dimensional set of input data, and $y_i$ the corresponding target variable. The network is optimised using ADAM optimisation [93], while the number of training epochs is determined through Early Stopping applied to the validation dataset (see Section 8.1.2 in [3]), tracking the validation loss with no minimum change requirements. We recognise that by using a validation set containing only 20% of the original training set, we may be severely limiting the number of points in the increasingly divergent regions, thus skewing our Early Stopping criteria to the less divergent regions. In an attempt to mitigate this, we train with a patience of 100 epochs to measure effects in the loss function significantly later in the training regime; however, at NLO we found that this makes minimal difference to the total loss and so can be reduced to speed up network training.

The inputs to the network are the 4-momenta of $n-1$ jets. Since we fix the centre-of-mass energy for training, we sought to reduce the number of input nodes for more efficient learning. We note that further reductions in the number of input parameters could be made, yet in testing this had no significant effect on performance.

### 5.2.3 Uncertainty Analysis

The subject of error and uncertainty analysis in ML processes is receiving increasing attention (see [162, 163] and the references therein), especially in the particle physics community [164–167], yet too frequently a demonstration of rigorous error analysis in ML regression processes is lacking.

As stated in [162], the main sources of error arise from approximation, *aleatoric* and *epistemic* uncertainties. Approximation uncertainty arises due to the model being too simplistic to allow for complex functional fitting, e.g. too few nodes or hidden layers in a neural network meaning the model is not able to fit sufficiently non-linear functions. Aleatoric uncertainty accounts for fluctuations in the data distribution e.g. from measurement errors, and cannot be decreased by collecting more data from the same experimental setup. Epistemic uncertainties, on the other hand, account for uncertainties in the model, including lack of sufficient coverage of the data. Since we are using deep neural networks, we assume the approximation error to be negligible. Additionally, we do not consider aleatoric uncertainties here since our data has been generated through high-precision numerical methods, and NJET accuracy tests have been performed to measure the stochasticity in matrix element generation and found this fluctuation to be negligible. [1] Following [164] we apply similar methods highlighted for use in classification networks to this regression

[1]NJET accuracy tests are performed by inferring on each phase-space point twice and checking the difference in the results. The threshold is set to the default value of $10^{-5}$ and errors arise due to lack of floating point precision and rounding errors.

task. Specifically, we focus on the measurement of precision/optimality errors, which include those arising due to epistemic uncertainties.

We measure model parameter initialisation dependence by training an ensemble of models on fixed training datasets while randomly reinitialising the weights of each model. Depending on the observable, the standard deviation in the bins can be measured. Additionally, when sampling the phase-space, the Monte Carlo error is calculated; however, this does not fully account for the uncertainty in phase-space completeness. For this, we bootstrap the training data, thereby resampling the phase-space multiple times and training an ensemble of models, with each model trained on a different dataset, while keeping the weight initialisations fixed. Since in this Chapter we are comparing neural network output against NJET results, we only include Monte Carlo errors on the NJET results, which avoids the double counting of errors. When using models 'in production', Monte Carlo error can be added to the model uncertainty, as specified above, for a full uncertainty estimate. We note that the best possible achievable accuracy would correspond to the Monte Carlo error on the NJET result.

The performance of our methodologies is also dependent on the test set chosen. For this we quote the Monte Carlo error, although it should be noted that the same issue with determining sampling completeness occurs here. Due to the computational expense of repeated generation of test sets, we do not perform this. However, the uncertainty bands on the neural network approximations should be sufficient to provide evidence of our methodology. This is because these additional dataset dependencies are negligible due to the large number of test points used and the relative size of the computed Monte Carlo error compared to the model uncertainties (see Section 5.3).

The errors on the models that we calculate are therefore the error due to model initialisation dependence and error due to the size of the training dataset, which are added in quadrature. As noted in [164], additional sources of uncertainty are inherent in the network approximation which are hard to calculate explicitly, such as dependence on the model architecture (e.g. the number of hidden layers, nodes in each layer and the types activation functions used). Due to the size of the other errors mentioned, and the lack of currently available tools for their calculation, we do not attempt to incorporate errors arising from these uncertainties into our analysis. We quote Monte Carlo error only for the testing dataset, with the exception for the NLO 5-jet case in which we quote both the Monte Carlo and model errors (see Figure 5.9).

When presenting our results, we calculate the mean of the ensemble of models trained and quote the standard error on the mean. Throughout this Chapter, we choose to train 20 models for each ensemble. This number was chosen in a slightly *ad hoc* manner, since it gave a reasonable distribution of models, and should not be interpreted as a requirement.

Theoretical uncertainties are also prevalent in all of these calculations due to variability in setting the renormalisation scale, $\mu_R$. Such uncertainties propagate through the networks since a model will learn to fit data at a certain scale. In this Chapter we train on data generated at a fixed scale, as used in [159], and we are not trying to teach the model anything about the scale uncertainties. However, to test that we are robust to different scale choices, we perform the normal *ad hoc* scale variation of $\mu_R/2$ and $2\mu_R$ purely to determine the dependence of our methodology on such a scale choice. In doing so we found that the models are able to approximate the matrix elements at each scale equally well to within Monte Carlo error, and we therefore assert that model performance is not highly dependent on the value of $\mu_R$ in the range we analysed. Moreover, since the goal of this work is not to calculate the cross-section or k-factors of a new processes, but to provide tools for estimating such values for already known process, we do not quote these as uncertainties in our methodology.

## 5.3 Results

We test our methodology on estimating both LO cross-sections and k-factors for processes up to $e^+e^- \to 5$ jets. In addition, various differential distributions are plotted to demonstrate the applicability of our methodology to real phenomenological calculations. In general, we see that neural network approximations demonstrate wide applicability to the cases investigated, with the FKS partitioning method giving more accurate and stable results through better approximations of the IR singularities. It should be noted that the cross-sections discussed here are calculated my summing the squared matrix elements and normalising by the number of phase-space points. This is a simplification of the partonic cross-section given in Equation 3.1.2.

### 5.3.1 Approximations at LO

Although leading order calculations are not significantly computationally expensive, they pose interesting test cases for neural network approximations of high multiplicity processes with many scales and complex IR singularity structures. Moreover, we find that much of what can be learnt from the performance of the models here can be applied to the NLO case.

As detailed above, we compare the naive approach where a single network is trained over the entire phase-space with the partitioned approach where an ensemble of networks trained on $N_{\text{max}} + 1$ partitions of phase-space. In determining the appropriate value of the global phase-space cut parameters, $y_{\text{cut}}$, we evaluate the performance of our models by calculating the ratio of the output to the NJET calculation as well as the model's ability to approximate the cross-section and differential distributions.

Figure 5.3 shows the distribution of the neural network errors by calculating the ratio of the model output to the NJET result at each phase-space point in the test set. Symmetric error distributions are desirable given that phenomenologically relevant

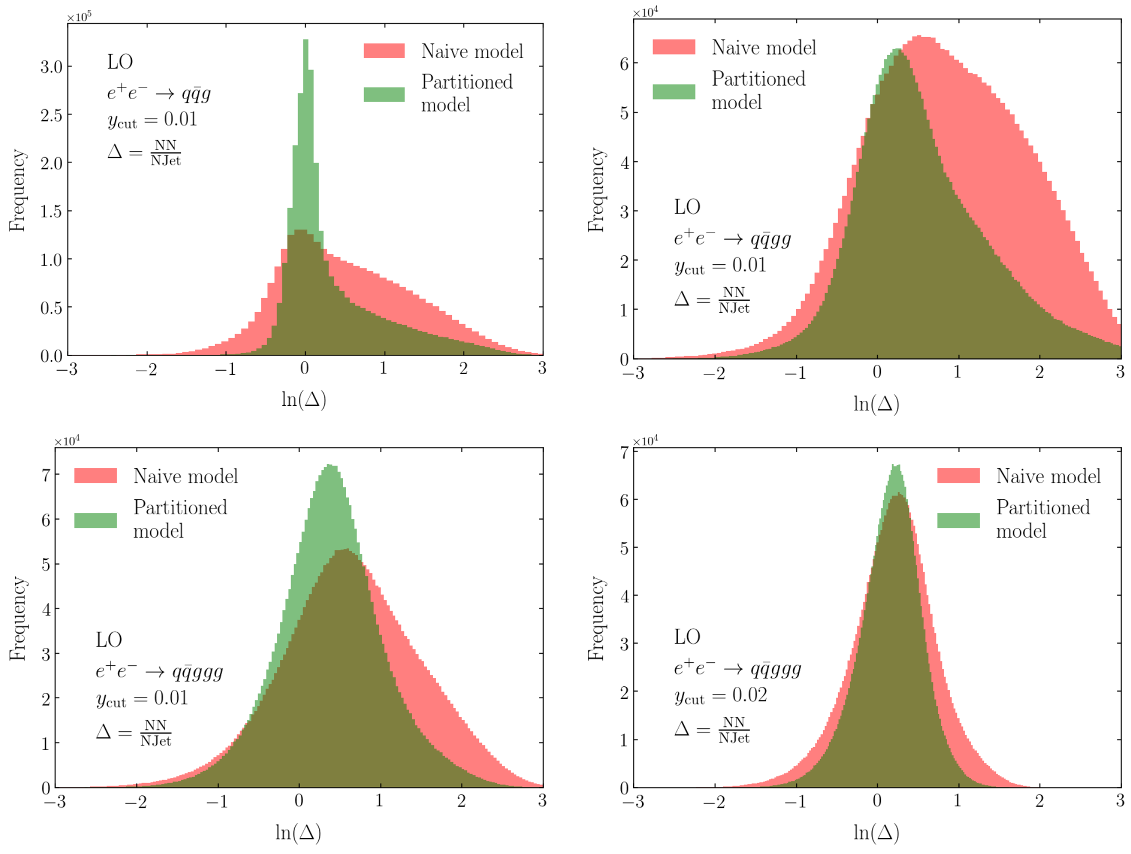


Figure 5.3: Born matrix element output of the naive approach (red) and partitioned approach (green) compared to the NJet calculation at different jet multiplicities and/or $y_{\text{cut}}$ values across 1M points. Outputs are taken as the average over 20 trained models.

tests for our method include approximating cross-sections and differential cross-sections. Since the partitioned approach gives much narrower and more Gaussian shaped distributions than the naive approach, we can clearly see that this method is preferable at the level of per-point accuracy. Additionally, the error distributions of the partitioned approach are more closely centred on zero in comparison to the naive approximation, thus suggesting that the partitioned model will produce a better overall average performance as well. The 5-jet error distribution from the partitioned approach appears to be narrower than the 4-jet, however, this is likely due to more points falling into the divergent region and providing the networks there with a greater amount of training data. Since the $y_p$ value has not been carefully tuned, this is not necessarily surprising. For more details on changing $y_p$ with multiplicity, see Appendix E. Note that these plots to not contain any information about the relative uncertainties attached to these model outputs, which we will discuss below.

While the error plots demonstrate the per-point performance of the models, we also wish to compare their performance in calculating physics observables while also taking into account uncertainty in the data and the model setup. Figure 5.4 shows the approximated cross-sections of the naive and partitioned approaches as compared to those computed from the NJET matrix elements. As expected, we see a harsher $y_{\text{cut}}$ value at 5-jets better regulates the divergent regions, thus improving both the naive and partitioned approaches; however, this harsher cut is not fully necessary as the NJET result sits on the edge of the neural network uncertainty bands.

When approximating the cross-section, we find the uncertainty bands have very little noise and follow the shape of the average result closely. Since each trained network will aim to minimise the value of the loss function, and no network will perfectly learn the target distribution, for each model there will be an offset between the final trained model result and the true distribution result. Since the cross-section is proportional to the average over the phase-space, for any value of $N$, these differences will average out such that the offsets manifest themselves as a distance away from the cross-section as calculated by NJET

$$\frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i) = \sigma_P - \sigma_{\text{NJET}} \tag{5.3.1}$$

$$= \epsilon + \mathcal{O}(\theta), \tag{5.3.2}$$

where $\sigma_P$ and $\sigma_{\text{NJET}}$ are the inferred and NJET calculated cross-section values respectively, $\epsilon$ is a fixed offset from the true cross-section and $\theta$ a small noise parameter. This therefore explains the relatively fixed distance between the model uncertainty upper and lower bounds and the NJET result.

Another result of Equation 5.3.2 is that, unlike Monte Carlo error, inferring on more test points will not reduce the model uncertainties since such a model cannot contain more information than the training dataset has provided. These uncertainties are intrinsically tied to the training set and the model initialisation and so any efforts to reduce errors arising at test time should therefore be focussed on addressing such uncertainties. We demonstrate an example of this by developing our partitioned method rather than focussing on changes to the test dataset.

In general, the global cuts required for the partitioned approach to be within the Monte Carlo error of the NJET cross-section are $\sim y_{\text{cut}} = 0.01$. These cut values are reasonable for our definition of $y_{ij}$ and are equivalent to the cuts made in [98].

After cuts have been made, we see that the partitioned approach has a significantly reduced standard error when compared with the naive approach, with an inferred mean closer to the final stable cross-section. This difference in uncertainty can be understood by comparing the relative standard deviations of the naive model's single network, and the deviations in the different networks making up the partitioned model, as we shall now show.

Let us first assume that the values of the cross-section calculated using the naive approach, $\sigma_s$, are normally distributed,[1] i.e. $\sigma_s \sim \mathcal{N}(\mu_s, \zeta_s^2)$, where $\mu_s$ is the mean of

[1]This is a reasonable assumption given that we would expect the uncertainty due to initialisation and dataset size to focus around a central mean value, with greater degrees of fluctuation becoming increasingly less likely. Additionally, any difference between the mean and the NJET result

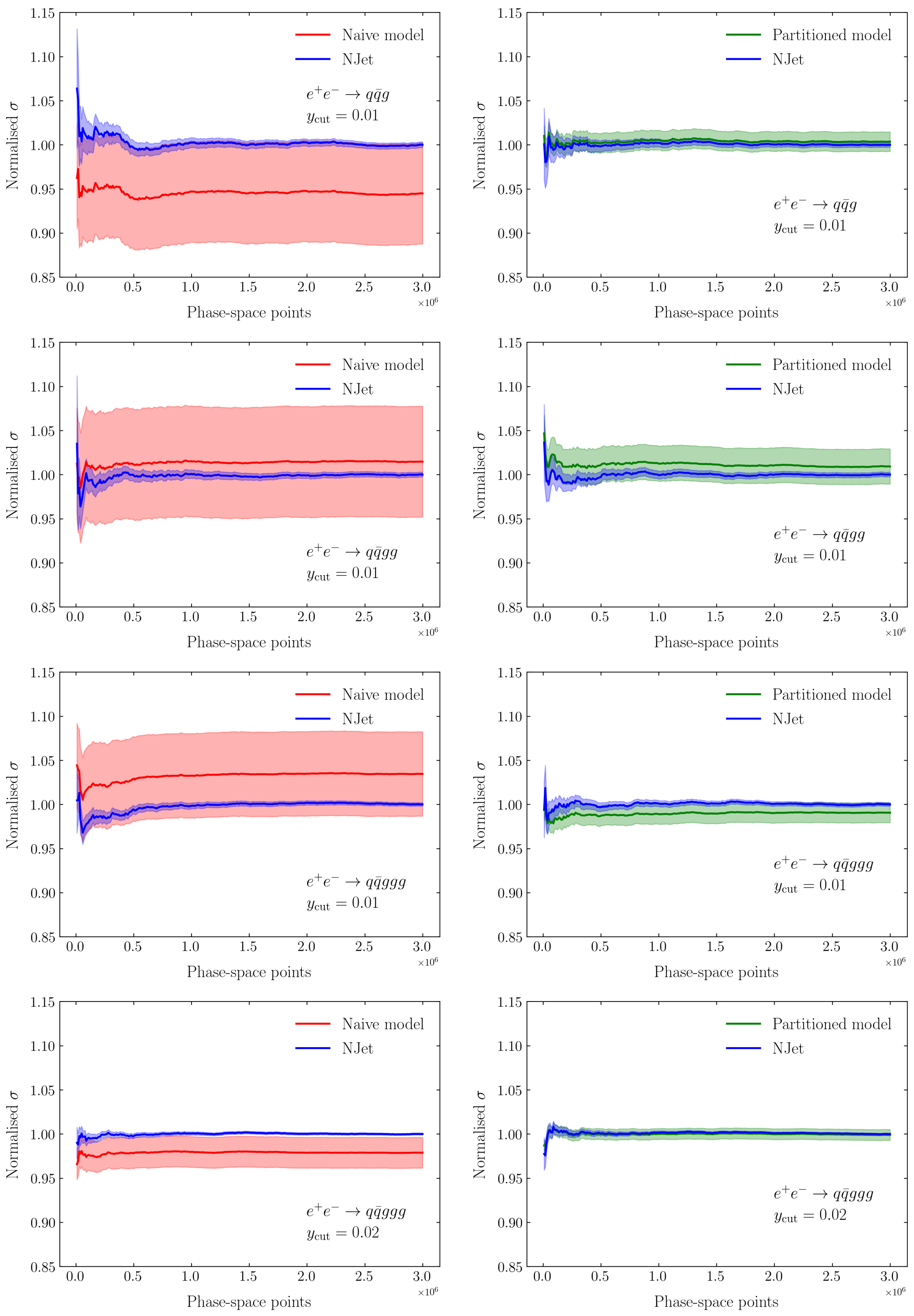


Figure 5.4: Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the Born normalised cross-section. Uncertainty bands denote the standard error on the mean calculated over 20 trained models (red and green) and Monte Carlo error on the NJET result (blue). We refer the reader to Section 5.2.3 for details of the error analysis.

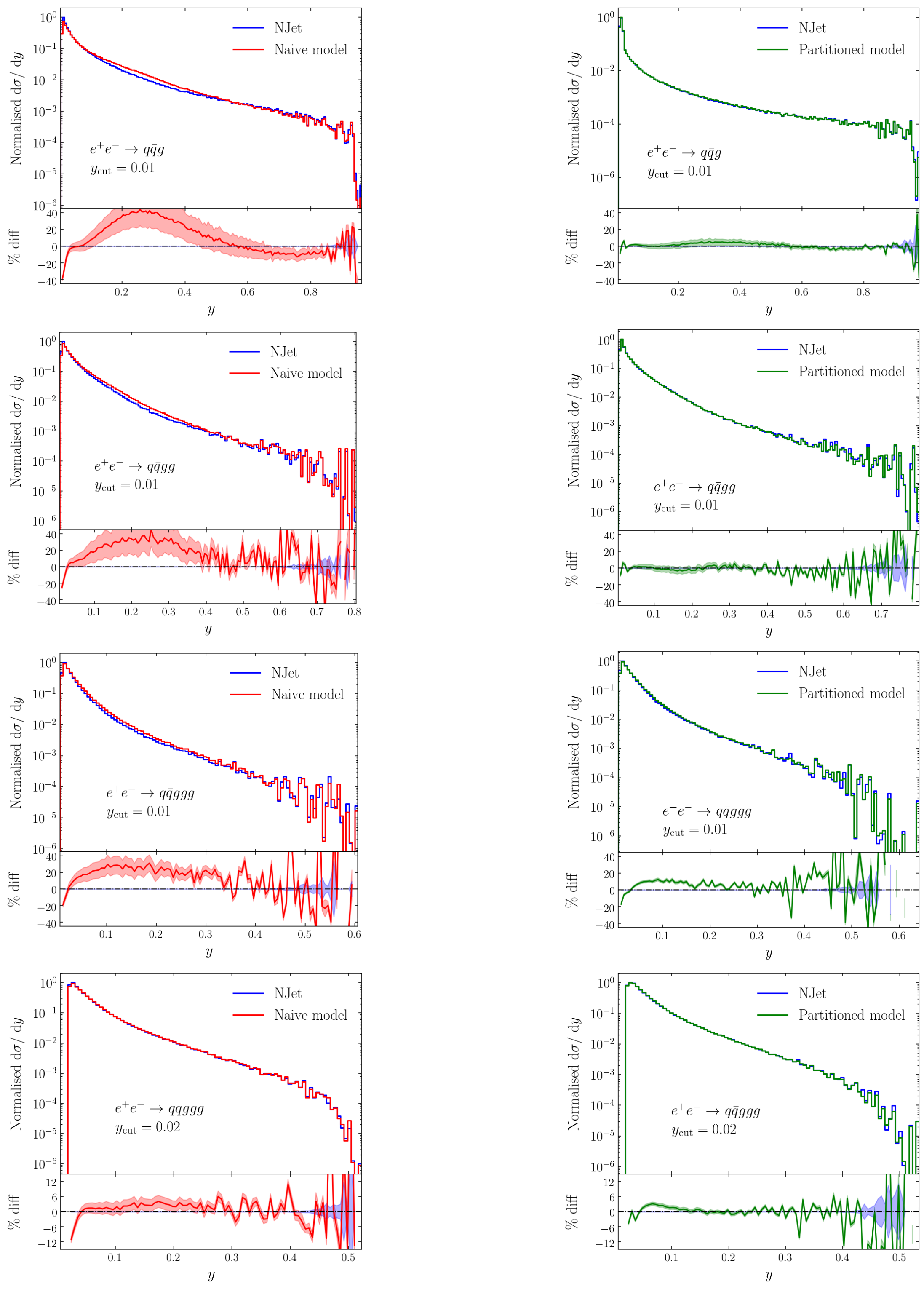


Figure 5.5: Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the differential cross-section against $y$, where $y$ is the minimum $y_{ij}$ as ordered by $p_T$. Data is normalised to the maximum NJet bin value. Uncertainty bands as described in Figure 5.4.

the normal distribution and $\zeta_s$ is the standard deviation. Secondly, we note that in the case of the partitioned method, the outputs of the networks trained over different partitions are first summed (c.f. Equation (5.2.6)) giving

$$\sigma_{\text{FKS}} = \alpha \sum_{p=1}^{N_{\text{max}}} \mathrm{d}\sigma_p + \beta\sigma_{\text{non-div}}, \tag{5.3.3}$$

where $N_{\text{max}}$ is defined in Equation 5.2.7, [1], $\mathrm{d}\sigma_p$ is the sum over all weighted matrix elements for a given FKS pair, and $\alpha$ and $\beta$ reweight the contributions of the local cross-sections in proportion to the number of total points in each region. Since we only partition the divergent region, $\mathcal{R}_{\text{div}}$, according to the FKS partition function, we add the differential cross-section over the non-divergent region, $\sigma_{\text{non-div}}$.

Given that the uncertainties in the individual networks making up the partitioned model are expected to manifest themselves in a similar way to the naive approach, we may also assume that these are drawn from a normal distribution such that

$$\forall p \in \{1, ..., N_{\text{max}}\} : \mathrm{d}\sigma_p \sim \mathcal{N}(\mu_p, \zeta_p^2), \quad \mathrm{d}\sigma_{\text{non-div}} \sim \mathcal{N}(\mu_{\text{non-div}}, \zeta_{\text{non-div}}^2), \tag{5.3.4}$$

$$\begin{aligned}
\implies \sigma_{\text{FKS}} &\sim \alpha \sum_{p=1}^{N_{\text{max}}} \mathcal{N}(\mu_p, \zeta_p^2) + \beta\mathcal{N}(\mu_{\text{non-div}}, \zeta_{\text{non-div}}^2) &(5.3.5)\\
&\sim \mathcal{N}\left(\sum_{p=1}^{N_{\text{max}}} \alpha\mu_p, \sum_{p=1}^{N_{\text{max}}} \alpha^2\zeta_p^2\right) + \mathcal{N}(\beta\mu_{\text{non-div}}, \beta^2\zeta_{\text{non-div}}^2) &(5.3.6)\\
&\sim \mathcal{N}\left(\sum_{p=1}^{N_{\text{max}}} \alpha\mu_p + \beta\mu_{\text{non-div}}, \sum_{p=1}^{N_{\text{max}}} \alpha^2\zeta_p^2 + \beta^2\zeta_{\text{non-div}}^2\right) &(5.3.7)\\
&:= \mathcal{N}(\mu_{\text{FKS}}, \zeta_{\text{FKS}}^2). &(5.3.8)
\end{aligned}$$

Since the uncertainties in the partitioned method are smaller than those found when using the naive approach

$$\zeta_{\text{FKS}}^2 < \zeta_s^2 \tag{5.3.9}$$

would likely be systematic of the model architecture choice, sampling algorithm and other factors external to the uncertainty measured here, thus resulting in a symmetric distribution, up to an approximation.

[1] In our implementation, for future process independence and coding simplicity we actually have $N_{\text{max}} + 1$ pairs since we do not discard the $q\bar{q}$ pair. In the processes examined in this Chapter, this has the effect of splitting the non-divergent region into two parts although, given the ease with which the networks are able to learn this region, we do not find this causing an issue.

$$\implies \alpha^2\zeta_p^2 < \zeta_s^2, \forall p \in \{1, ..., N_{\text{max}}\} \text{ and } \beta^2\zeta_{\text{non-div}}^2 < \zeta_s^2. \tag{5.3.10}$$

From Equation 5.3.10 we see that not only does the partitioned method have a reduced uncertainty in comparison to the naive method, but that each individual network making up the partitioned model also has a reduced uncertainty contribution, thus supporting the claim that by using the partitioned method, the networks learning the divergent structure are more certain, relative to their contributions, about what they are learning and less sensitive to both model initialisation and dataset size.

The overall accuracy of the partitioned approach, combined with the implications of Equation 5.3.10, demonstrates that we are learning the divergent structure of the amplitude sufficiently well. As discussed in Section 5.2.3, it should be noted that Figure 5.4 and Figure 5.5 do not show the performance of a single model, but rather the average of 20 trained models with their equivalent standard error. Although one does not have to train this many models to get a good approximation, in Section 5.3.2 we will see that training additional models is computationally cheap and thus not a large hinderance.

Figure 5.5 shows the differential cross-section of the $y_{ij}$ distribution of the two softest jets as ordered by $p_T$. Again, we plot the mean of the 20 trained models and the standard error on the mean. These differential distributions were chosen as they highlight the performance of the models in hard-to-sample regions of phase-space, in particular some of the regions we would expect the FKS partition function to assist with learning. Indeed, we see a significant improvement when using our partitioned method both in comparison to the performance of the naive approach, in overall per-bin, accuracy and in stability. In addition, the partitioned method also produces narrower uncertainty bands than the naive approach, thus demonstrating its higher confidence in these regions. While this confidence is seen to be slightly misplaced in the case of the 5-jet plot at $y_{\text{cut}} = 0.01$, we see the harsher cut mostly correcting for this and producing good agreement between the NJET and partitioned results. Similar reasoning to that given in Equations 5.3.4 - 5.3.10 can be applied to the per-bin uncertainty differences between the naive and partitioned approaches.

Overall, the partitioned model is shown to produce more accurate and reliable results in LO approximations than the naive approach. While it can be argued that there is greater computational expense in training multiple networks, given the very low cost of network training in comparison to the data generation time this is considered to be negligible, particularly at higher orders (see Section 5.3.2 for more details).

### 5.3.2 Virtual Approximations at NLO

When approximating the k-factor, the IR singularities present in the previous examples have been normalised by dividing out the LO matrix element. This normalisation regulates the number of large divergences in phase-space, allowing the network to focus more on learning the loop-induced divergences. Additionally, although the FKS method is especially useful for isolating soft and collinear divergences at LO, given the presence of logarithmic terms in $s_{ij}$ in the virtual corrections, we still expect to see improvements by using the partitioned method when approximating the k-factor.

As in the LO case, in Figure 5.6 we plot the error distributions for the naive and partitioned cases by comparing the network outputs to the NJET calculations at the per-point level. In the 3 and 4-jet cases we see that both methods perform relatively similarly, with the naive approach appearing to be slightly better in the case of 4-jets. However, it should again be noted that these plots do not contain information about the network uncertainty and so should not be interpreted as the sole measure of performance.

In Figure 5.7 we see that both the naive and the partitioned approaches approximate the k-factor to within Monte Carlo error at 3-jets, and are within the percent level at 4-jets. Although either methodology would be suitable for use, the partitioned

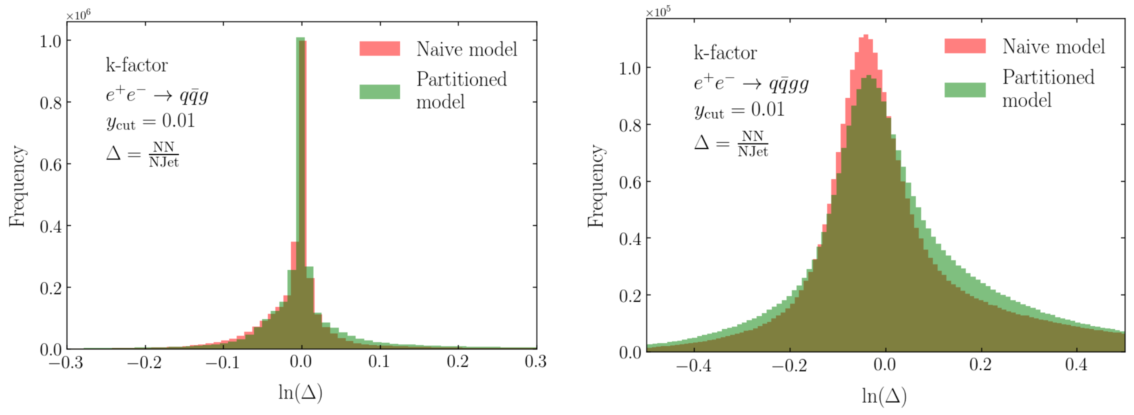


Figure 5.6: k-factor output of the naive approach (red) and the partitioned approach (green) compared to the NJET calculation at different multiplicities. Outputs are taken as the average over 20 trained models.

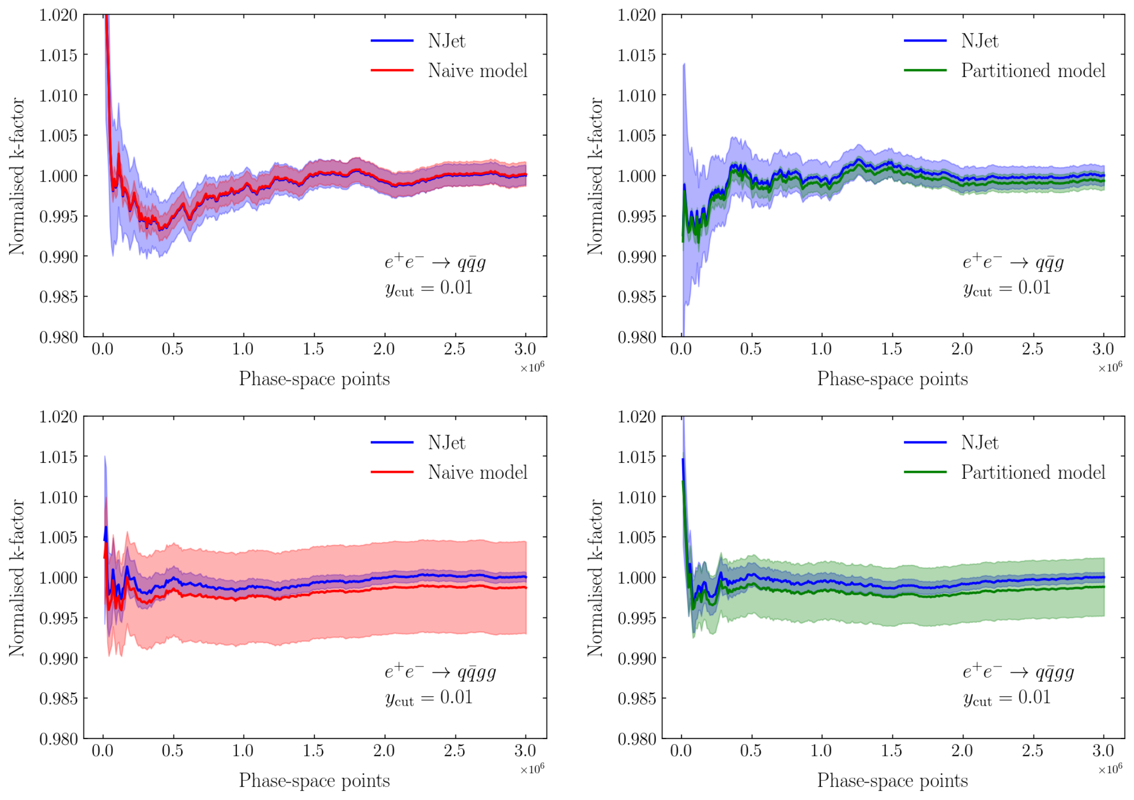


Figure 5.7: Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the normalised NLO/LO k-factors. Uncertainty bands as described in Figure 5.4.

approach requires little more computational time in comparison to the naive model, while producing narrower uncertainty bands. For robustness at higher multiplicity, the partitioned method remains the more optimal method.

A comparison between the computational speed of different methods of k-factor computation and calculation can be found in Table 5.1. Here we see a dramatic speed-up when using the network approximation as opposed to current numerical methods, with the dominant time saving coming from the reduction of the number of matrix elements having to be explicitly calculated using NJET (i.e. in the case of training on 100k points and inferring on 1M at high multiplicity the speed-up is $\mathcal{O}(10)$). Moreover, the assertion that the partitioned method is not significantly more expensive than the naive approach can be verified. It should be noted, by only training on 10k points we may achieve unacceptable performance when compared to the 100k results. The results presented in the table are therefore designed to demonstrate the computational time required for network training in comparison to the NJET calculation, as opposed to providing guidelines on how many training points to use.

As in Section 5.3.1, we plot the differential k-factors of the $y$ distribution of the two softest jets as ordered by $p_T$. In Figure 5.8 we see that both the naive and partitioned approaches model the data well. As before, the partitioned method provides us with slightly narrower uncertainty bands in both the 3 jet and 4 jet cases. Additionally, although neither the naive model, nor partitioned model, approximate the peak in the 4 jet distribution exactly, the peak location is more accurately approximated by the partitioned approach with only a single bin at the peak being significantly ill-approximated. While we do not necessarily see much improvement in using the partitioned approach, given that the additional training time required is negligible in comparison to the data generation, as well as its performance in approximating the overall cross-section, we still see the partitioned approach as a viable and beneficial method to use for k-factor approximation. It should be noted that similar reasoning as given in Equations 5.3.4 - 5.3.10 can again be applied to the k-factor and per-bin

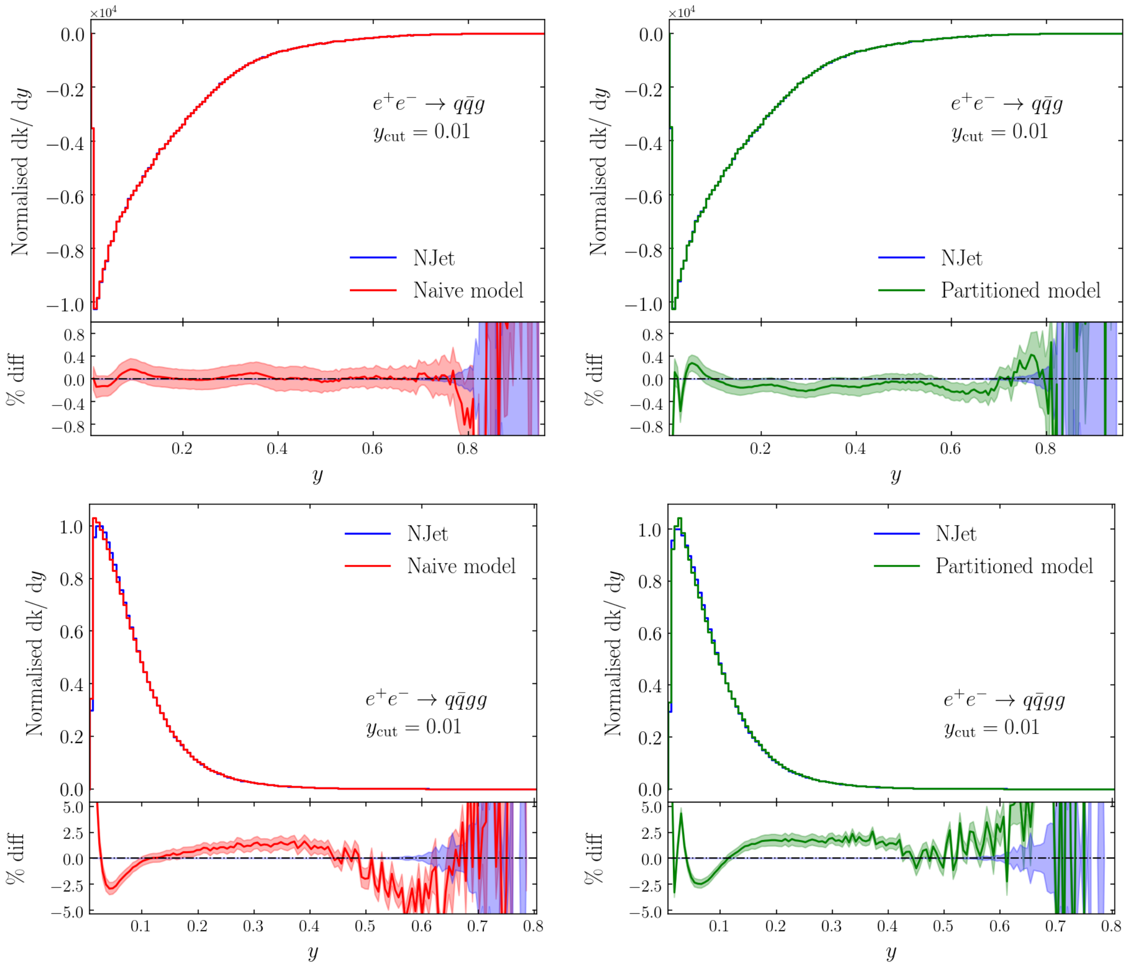


Figure 5.8: Comparison of the naive approach (left) vs. the partitioned approach (right) in estimating the differential NLO/LO k-factors against $y$, where $y$ is the minimum $y_{ij}$ as ordered by $p_T$. Data is normalised to the maximum NJet bin value. Uncertainty bands as described in Figure 5.4.

| | 10k training, 1M inference | | | 100k training, 1M inference | | |
|---|---|---|---|---|---|---|
| | NJET | Partitioned approach | | NJET | Partitioned approach | |
| Jets | Time (hrs) | Time (hrs) | % diff | Time (hrs) | Time (hrs) | % diff |
| 3 | 13.2 | 0.15 | $-0.5 \pm 0.3$ | 13.2 | 1.32 | $0.1 \pm 0.2$ |
| 4 | 194 | 1.97 | $0.5 \pm 0.5$ | 194 | 19.4 | $0.1 \pm 0.4$ |
| 5 | $6.39 \times 10^3$ | 63.9 | - | $6.39 \times 10^3$ | 639 | - |

Table 5.1: Time required for k-factor calculation at different multiplicities requiring 1M points, while training on 10k and 100k points. Performance of the partitioned approach is assessed by calculating the percentage difference in the cross-section approximation normalised to the NJET result. Errors are calculated by adding the model uncertainty and Monte Carlo error from the NJET result in quadrature. These results assume all calculations take place on a single CPU core and that the training points form part of the inference set. Training on 10k points is fast but not necessarily reliable, whereas using 100k points gives more reliable results and so may be a more reasonable estimate of the speed-up. Results are not given for 5-jets since we did not generate testing data at this multiplicity.

uncertainty differences between the naive and partitioned model approaches at NLO.

Finally, in the case of 5-jets we demonstrate our methodology as it may be used in practice. In Figure 5.9 we show how one may infer on a set of points with no known NJET results for testing, while understanding the associated neural network errors. From these plots we clearly see that the partitioned method has associated errors only at the level of 0.5% in the cross-section, with larger uncertainties in the regions of the differential plot where one would expect Monte Carlo error to dominate.

As highlighted above, when you do not have a test set for comparison, it may be hard to validate the optimal number of training points required for a good approximation. While at NLO we present the results of networks trained on 100k points, and found this number to be relatively optimal with regards to accuracy, stability, and training time, we do not claim that this will always be the case for other processes. Although generating more NJET matrix elements for testing is the best way to assess network accuracy, a possible substitute would be to test on the training data. While this

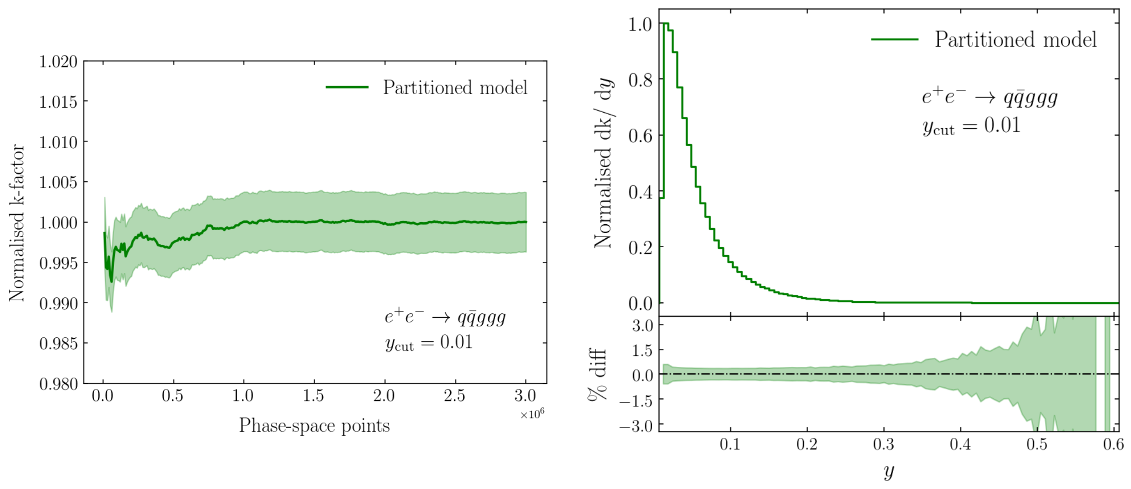


Figure 5.9: Normalised NLO/LO k-factor and differential k-factor against $y$, where $y$ is the minimum $y_{ij}$ as ordered by $p_T$, at 5 jets using just the partitioned approach. Data in the differential plot is normalised to the maximum network output value. Uncertainty bands denote the following errors added in quadrature: one standard error from model uncertainties and one Monte Carlo error on the result itself. Uncertainty bands are given as a percentage of the mean calculated over 20 trained models.

is not generally regarded as good practice, given the problem at hand it may not be as bad as in other cases. For instance, unless there is a large degree of noise in the cross-section given the size of the training dataset, as an initial measure of model performance we can quantify the uncertainty in our training set and assess the proximity of our network uncertainties and this Monte Carlo error. Additionally, our network uncertainty calculation depends only on the network's behaviour relative to the training set and is independent of the test set. Therefore, although testing on the training set is still not ideal, given how we calculate our network uncertainties and by using our physics knowledge of the Monte Carlo error, we are able to use this as a first test of network performance without having to generate additional testing data.

## 5.4 Summary

In this Chapter we have explored the possibility of optimising simulations for many-scale processes needed for LHC analyses. Machine learning technology is finding an increasing number of applications in particle physics and offers the potential to dramatically reduce the CPU cost of expensive simulations.

The application to scattering amplitudes is a little different to classic examples of neural networks in that the dataset is exact. [1] We can also have complete control over the range of the dataset, although the CPU cost of obtaining the data can be very high. The challenge is to make a sufficiently good fit to the data that a reliable interpolation and extrapolation of differential cross-sections can be made. The CPU cost of the extrapolation/interpolation is negligible in this procedure so the further the network can be extrapolated, the better the computational speed-up.

In this Chapter, we have looked at multi-scale amplitudes which are not well suited to more traditional approximations with polynomial grids. At one-loop, scattering for $2 \to 4$ or higher multiplicity becomes extremely expensive, even with modern automated tools. We find that a reliable amplitude approximation can be difficult to achieve when using a naive single neural network due to the large changes in the amplitude related to its singularity structure. We compare this naive approach to a technique in which an ensemble of networks are used to approximate the amplitude by separating the singularities, using an FKS partitioning. This partitioned approach has shown promising results at both the per-point level as well as in estimating the cross-section and differential distributions.

Understanding the reliability of this approach is one of the biggest challenges. By varying the initial data and parameter initialisations used in the network, we find a way to estimate the error on the networks. For all but the highest multiplicity, $e^+e^- \to 5$ jets, we also provide comparisons to direct integration of the amplitude.

[1]Technically we restrict to double precision, although higher precision arithmetic could be used in principle.

At LO we observe that the FKS partitioning provides significantly more reliable and accurate estimates than the naive approach, while in the case of NLO k-factors, where the leading order singularity structure is divided out, the partitioning still helps in these regards, with results accurate to within a few percent. Moreover, Equations 5.3.4 - 5.3.10 show that each network in the partitioned model has a smaller associated uncertainty than that of the naive model, thus suggesting that the partitioned model is learning the divergent structure with a higher confidence than the naive model. Indeed, this is the case at both LO and NLO. The networks not only provide good scattering amplitude approximations, but also lead to reliable predictions with a speed up comparable to the ratio of the number testing to training points.

In this initial study we have made a number of simplifications whose effect could be important when using the technique for a realistic analysis. Firstly, we employed a simple flat phase-space generation using the RAMBO algorithm. This makes it hard to compare with the more efficient generators used in state-of-the-art Monte Carlo simulations. The JADE jet algorithm may exacerbate the soft singularities and so the effect of alternative jet algorithms, as well as the effect of introducing initial state singularities in *pp* collisions. We also see in the higher multiplicity cases that the error from the neural network approximation does start to increase. It may be in these cases that the NLO FKS separation requires modification. In this study we used a simple version of the partition function based only on the kinematic invariants. In general, we can alter the scaling power of the invariants in the various limits which will affect the behaviour of the FKS regions away from the singularities.

To address some of these concerns, in the next Chapter we explore the applicability of the methodology developed above, to increasingly complex gluon-induced diphoton scattering processes. Such processes introduce the challenges of initial-initial and initial-final state singularities, which were not present in $e^+e^-$ collisions, as well as the absence of tree-level diagrams, since these processes are loop-induced at LO. We will also assess how to integrate these ML techniques with existing Monte Carlo event generator technologies, such as Sherpa, and test the robustness of our ML approach to more phenomenologically relevant cuts.

# Chapter 6

# Machine learning for matrix element approximation: diphoton + jets

## 6.1 Motivation

In Chapter 5 we developed an ML pipeline for learning matrix elements for $e^+e^-$ collision processes at both tree and one-loop level and at high multiplicity. This provided a good test bed in which to develop our methodology, without having to be concerned about initial-state singularities, and additional factors such as PDFs which alter the centre of mass of the system. In this Chapter, we will extend this work to consider a class of scattering processes that contribute to diphoton signals at hadron colliders. We will also introduce a systematised pipeline for using the ML approach described previously for phenomenological studies in practice.

The process $gg \to \gamma\gamma + n(g)$ is a good test case for ML technology, since it is loop-induced and has relevant contribution from high multiplicity matrix elements. Using automated tools at NLO, full QCD corrections are known for $pp \to \gamma\gamma + \leq 3$ jets [168–170]. There has been a flurry of recent activity around next-to-next-to-leading-order (NNLO) corrections to $pp \to \gamma\gamma + j$ in which the complete leading colour corrections have been presented [171–175]. As such, $2 \to 3$ and $2 \to 4$ scattering for the gluon-initiated diphoton channel are now extremely relevant for future phenomenological studies.

The interface of general one-loop amplitude codes into multi-purpose Monte Carlo event generators has resulted in a wide variety of simulation options which can offer the best possible theoretical accuracy. Methods that go beyond fixed-order perturbation theory — such as parton shower matching, merging, and jet multiplicities — improve accuracy across important regions of phase-space. However, these simulations add additional strain on the underlying amplitudes.

State-of-the-art tools make use of advanced phase-space mapping algorithms to improve the convergence of the multi-dimensional integration (see Chapter 3). General purpose Monte Carlo event generators such as Sherpa [2,50], Pythia [45,46], Herwig [47–49], and MadGraph [51] often make use of the diagram structure of the underlying tree-level process to ensure an optimal distribution of points during the integration grid optimisation phase. Reusing tree-level distributions when generating virtual events is particularly effective at reducing the computational cost of using complicated one-loop amplitudes. However, this is not possible in the loop-induced processes discussed here.

Given these problems the $gg \to \gamma\gamma + n(g)$ processes present a relevant and present challenge to event generation. In this Chapter we will not only apply our pipeline to this process and explore its performance, but also interface our approach with Sherpa, and assess its robustness to the additional complexities which arise.

## 6.2 Gluon-initiated diphoton amplitudes

We study amplitudes with two photons and many gluons which first appear at one-loop level in the SM. With conventional simulations relying on cheaper LO tree evaluations to optimise event generation for NLO one-loop contributions, these loop-induced processes present an interesting sector to test new approaches for phase-space integration. Compact analytic computations for $gg \to \gamma\gamma$ and $gg \to \gamma\gamma g$ have been available for some time and offer extremely fast and stable evaluation. As a result, it is feasible to optimise event generation with the one-loop evaluation. For $2 \to 4$ scattering problems, only numerical codes are available and simulations can be extremely slow. It is also not clear that analytic formulae would be sufficiently compact to alleviate this situation, even if they were available.

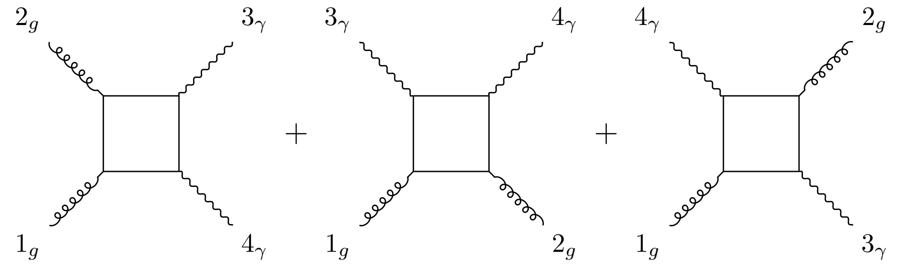
The loop-level amplitudes proceed through a fermion loop and have a colour decomposition in the trace basis as

$$\mathcal{M}^{(n-2,1)}(1,\dots,n-2,(n-1)_\gamma,n_\gamma) = \\ {g_s}^{n-2} g_{\gamma\gamma} \sum_{\sigma\in S_{n-3}} \lambda(\sigma\,(a_1,\dots,a_{n-2}))A^{(1)}(\sigma(1,\dots,n-2),(n-1)_\gamma,n_\gamma), \quad (6.2.1)$$

where $g_s$ is the strong coupling, $g_{\gamma\gamma} = g_e^2 \sum_q Q_f^2$ is the combined coupling of the diphoton system to the fermion loop, $S_{n-3}$ is the set of even non-cyclic permutations of $\{1,\dots,n-2\}$, $f$ runs over active quark flavours with fractional quark charge $Q_f$, and the colour trace function $\lambda$ is defined as

$$\lambda\,(a_1,\dots,a_{n-2}) = \mathrm{tr}\,(t^{a_1}t^{a_2}\dots t^{a_{n-2}}) + (-1)^n\mathrm{tr}\,(t^{a_1}t^{a_{n-2}}\dots t^{a_2})\,. \quad (6.2.2)$$

For example, for $n = 4$ there is a single primitive amplitude. [1] It is given by the diagrams

$A(1,2,3_\gamma,4_\gamma) =$  ,

(6.2.3)

where a plain line indicates a sum over quark loop arrow directions. At one-loop, these amplitudes are also related to the fermion loop corrections to pure gluon

[1]Primitive amplitudes are gauge invariant and have fixed cyclic order of external legs. The permutations, $\sigma$, introduce the sum over the different permutations of the QCD generators.

scattering through permutations [176]. The ingredients for differential cross-sections are the squared amplitudes summed over helicities, $h$, and colour,

$$|\mathcal{M}^{(n-2,1)}|^2 = \left(\frac{\alpha_s}{4\pi}\right)^{n-2} g_{\gamma\gamma}^2 \sum_{h,i,j} A_i^{(1)^*}(h)\, \mathcal{C}_{ij}\, A_j^{(1)}(h) + \mathcal{O}(\alpha_s^{n-1}) \tag{6.2.4}$$

where the matrix $\mathcal{C}$ is a function of the number of colours, $N_c$, obtained by squaring the colour basis elements, and the index on the partial amplitudes, $A$, refers to the different permutations in the colour decomposition.

As in Chapter 5, the amplitudes are taken from the NJet C++ library [65]. Here, there are different options: a general numerical setup using generalised unitarity and integrand reduction; and hard-coded analytic expressions for $n = 4, 5$. The $n = 4$ analytic expressions were taken from [177], while for $n = 5$ they were obtained directly from a finite field reconstruction [178] and are in agreement with known analytic formula [176, 179].

The numerical evaluation requires the sum of permutations of ordered primitive amplitudes. This is completely automated for arbitrary multiplicity, but evaluation times and numerical stability are increasingly difficult to control.

To study the growth of evaluation time with multiplicity, we evaluate the matrix element at 100 random phase-space points with each available technique and plot the mean times in Figure 6.1. We generate the phase-space points isotropically with the algorithm from [180]. While analytic methods are competitive at low multiplicity, we see they scale poorly and are unlikely to beat numerics at $n \geq 6$. Numeric scaling is better, but these algorithms come with a high cost. Our NN approach provides a performant alternative, with significantly better scaling than either numerics or analytics.

## 6.3 Computational setup

In this Chapter, we build on the work presented in Chapter 5, where a NN ensemble approach was presented in which a different NN is trained on each soft and collin-

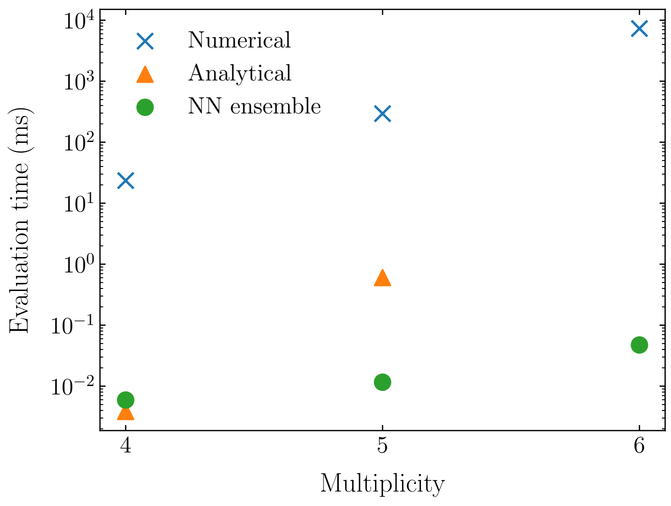


Figure 6.1: Matrix element typical CPU evaluation times for available methods — including NJET numerical evaluations, NJET analytical evaluations, and inference on a NN ensemble as described in Section 5.2 — against the number of legs. These calls are single-threaded as parallelisation is applied at the level of events in simulations. An analytic expression for $2 \to 4$ is not available. The NN is comparable to the analytic call at $2 \to 2$, 50 times faster at $2 \to 3$, then $10^5$ times faster than the $2 \to 4$ numeric call. All NN evaluations were performed using the same $y_p$ parameter value. The differences in evaluation time with increasing multiplicity is therefore a result of points falling into $\mathcal{R}_{\text{div}}$.

ear region of phase-space, and was shown to be effective in handling IR divergent structures at both the Born and one-loop level at high multiplicity in $e^+e^-$ collisions. In the case of the $2 \to 3$ and $2 \to 4$ gluon-initiated diphoton amplitudes, we have initial-initial, initial-final and final-final state IR singularities. Therefore, in this Section, we will present a generalisation of the phase-space partitioning introduced in Section 5.2.1 to hadron-hadron collisions as specified in [38]. We will also discuss the interfacing of our methodology with existing event generators such as Sherpa [2,50]. This is important to demonstrate since it is not immediately obvious that NN approximations trained in isolation will be robust to the added intricacies of event generators important for extracting physical results, such as PDF weightings and choices of integrators.

## 6.3.1 Phase-space partitioning for hadron-hadron collisions

In Section 5.2.1 we introduced our method for phase-space partitioning based on FKS subtraction [37,38] techniques. We now generalise the divergent and non-divergent regions of phase-space presented in Equations 5.2.2 and 5.2.3 to include the relevant additional FKS pairs

$$\mathcal{R}_{\text{div}} = \Big\{ p \,|\, \min(y_{ij}) \leq y_p,\, p = (p_1, p_2, \ldots, p_n),\, i,j \in \{1, \ldots, n\} \Big\}, \qquad (6.3.1)$$

$$\mathcal{R}_{\text{non-div}} = \Big\{ p \,|\, y_p \leq \min(y_{ij}),\, p = (p_1, p_2, \ldots, p_n),\, i,j \in \{1, \ldots, n\} \Big\},. \qquad (6.3.2)$$

In Equations 6.3.1 and 6.3.2, we have removed the $y_{\text{cut}}$ parameter introduced in Section 5.2 since we will be using more phenomenologically relevant global cuts in this Chapter (see Section 6.3.2). The FKS pairs are then defined as

$$\begin{aligned}\mathcal{P}_{\text{FKS}} = \{(i,j) \mid 1 \leq i \leq n,\ 2 \leq j \leq n,\ i \neq j, \\ \mathcal{M}^{(n,0)} \text{ or } \mathcal{M}^{(n,1)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i \parallel \vec{p}_j\}. \quad (6.3.3)\end{aligned}$$

The partition functions are then the same as in Equation 5.2.5, with the newly defined FKS pairs from Equation 6.3.3.

The above definitions are appropriate for the processes studied in this Chapter as they account for the different singularity structure to that found in the case of $e^+e^-$ (i.e. they explicitly include the initial-state singularities). To allow for easier generalisability to other processes, we include all pairs of initial- and final-state particles in our implementation, including the $\{\gamma\gamma\}$ pair which is redundant as it does not exhibit the relevant singularity structure. This redundancy does increase the computational time required; however, we find the performance of the NN ensemble is not adversely affected. The above implementation could therefore be simply generalised to the $e^+e^-$ case, although again with some redundancy.

### 6.3.2 Neural network setup

While the focus of this Chapter is the use of the NN ensemble method presented in Chapter 5, for completeness we present a comparison of this method against a naive (single network) approach in Appendix F. The same network architecture as presented in Section 5.2.2 is used. Hyperparameter tuning also comfirmed the appropriateness of this choice (see Appendix D for more details).

#### Data

The sampling of phase-space is dependent on the integrator. In this Chapter we use integrators already available in Sherpa. Unless otherwise specified, the same integrator is used for training, validation and testing. We generate the datasets from two runs of the integrator: the first is divided into training and validation datasets according to an 80:20 split; the second uses a different random seed from the first, and is used for the test dataset (with the exception of error plots showing the per-point accuracy of the models). Otherwise, data is generated as in Section 5.2.2 for the NLO case, and we train our networks on 100k points and test on 3M. [1]

[1]Varying types of data processing were used for hyperparameter tuning (see Appendix D for more details). However, standardisation was still the chosen method.

All our simulations use $\sqrt{s_{\text{com}}} = 1\,\text{TeV}$; the methodology is theoretically agnostic to this choice. In Chapter 5, a simple JADE algorithm was used for the global phase-space cuts. To test the robustness of our approach to more realistic phenomenological cuts, unless otherwise specified all models are trained, and analyses performed, using the following kinematic cuts adapted from those in [169]

$$\begin{aligned}
&p_{T,j} > 20\,\text{GeV} \qquad && R_{\gamma,j} > 0.4 \qquad && \left|\eta_j\right| < 5 \\
&p_{T,\gamma_1} > 40\,\text{GeV} && R_{\gamma,\gamma} > 0.4 && \left|\eta_\gamma\right| < 2.37 \\
&p_{T,\gamma_2} > 30\,\text{GeV} && &&
\end{aligned}$$

where $p_T = \sqrt{{p_x}^2 + {p_y}^2}$ (beam along $z$-axis) is transverse momentum magnitude, $R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ is isolation cut cone radius, $\eta$ is pseudorapidity, $\phi$ is azimuthal angle, $\gamma$ denotes a photon, photons are ordered by $p_T$, and jets, $j$, are identified through the anti-$k_T$ algorithm [181] implemented in FastJet [182] with $R = 0.4$. These cuts are typical for LHC analyses. Photons are selected by smooth cone isolation [183] such that all cones of radius $r_\gamma < R$ satisfy

$$E_{\text{hadronic}}(r_\gamma) \leq \epsilon\, p_{T,\gamma} \frac{1 - \cos r_\gamma}{1 - \cos R}$$

with $R = 0.4$ and $\epsilon = 0.05$.

Matrix elements are evaluated with renormalisation scale $\mu_R = M_Z$, where $M_Z$ is taken from the PDG [1]. Since the one-loop process is LO, the full amplitude is finite and has $\mu_R$ dependence in the couplings only.

### 6.3.3 Interfacing with event generators

Assessing performance after interfacing with existing event generator technology is important for demonstrating ‘real-world deployment’ of ML algorithms in particle physics simulations, as it exposes the model to a range of post-inference effects which may alter the final reliability of the model. For example, generators allow for the easy implementation of complex phase-space cuts, jet clustering algorithms, phase-space and PDF weights, as well as different integrators and integration optimisation routines.

### The interface

Event generators are largely written in C++ for computational efficiency. Therefore, after the model has been trained, the weights of each NN are extracted and written to file. A C++ program reads these models' files and performs the linear algebra operations required during the inference step, using EIGEN [184]. This means the PYTHON libraries used for model training and inference are circumvented, and the call time for model inference is reduced, while keeping everything in C++ simplifies the interfacing of the model with standard event generators.

Given a set of 4-momenta, a custom C++ interface provides the helicity- and colour-summed matrix element to SHERPA. This can be used to call NJET evaluations through a BLHA interface [155,156] or to call the model inference result. RIVET [185, 186] is then employed for analysis, with a script adapted from the reference analysis of [187].

### Phase-space integration

Phase-space integrators seek to achieve increasingly optimal rates of integration convergence through the careful sampling of points. While the choice of integrator can affect the overall rate of convergence, it also determines the placement of phase-space points which directly feeds into the distribution of points in the training dataset.

Since these processes are loop-induced, for simplicity we use the RAMBO integrator [6] throughout for event generation (see Appendix B.1). However, we test different approaches to generating the integration grid. The first we term the 'unit grid', which is constructed by running the grid optimisation step while returning a unit value in place of the matrix elements. This effectively removes the dependence on the optimisation procedure and, since RAMBO is used, ensures a uniformly and isotropically sampled phase-space. The second uses VEGAS [73, 74] optimisation when generating the integration grid, thereby putting a preference on sampling regions of particular importance to the cross-section (see Appendix B.2). We share the integration grid between training and testing phases, meaning this importance sampling is reflected in both. Given the expense of matrix element calculation for the $2 \to 4$ scattering process, we reserve the use of VEGAS optimisation only for the $2 \to 3$ case.

**Weights**

When training the models, we do not include explicitly any event generator effects. All additional weightings, i.e. phase-space weights and PDF weights, are introduced after the model has been used for inference, as is done for other matrix element generators. The addition of these weightings has the potential to be problematic for model performance: when the model is trained it is unlikely to learn all regions of phase-space equally well and there is a chance that those regions in which the model has poor performance could be amplified by these additional weighing factors.

In order to test for this we include PDF weights using the LHAPDF library [188] and the NNPDF3.1 set `NNPDF31_nlo_as_0118` [189] as well as phase-space weights which depend on the integration grid optimisation method.

### 6.3.4 Reweighting

The approach used in this Chapter to train the NN ensemble provides good agreement between the network output and that of NJet. However, the ensemble approach will always be an approximation and is subject to perform poorly in certain regions of phase-space, especially those in which it has not been trained or in which training data are sparse. As a partial remedy to this, we propose the idea of reweighting the event weights with known matrix element values derived from NJET. When using weighted event generation, this can either be performed after event generation or can be done ‘on the fly’ at the interface level. The former approach is possible since the original event weight can be recovered through

$$w_{E,i}^{(\text{NJET})} = w_{E,i}^{(\text{NN})} \times \frac{\mathcal{M}_i^{(\text{NJET})}}{\mathcal{M}_i^{(\text{NN})}}, \tag{6.3.4}$$

where $w_{E,i}^{\text{NJET}}$ and $w_{E,i}^{\text{NN}}$ are the event weights using NJET and the NN ensemble respectively for a given phase-space point $i$, and $\mathcal{M}_i^{\text{NJET}}$ and $\mathcal{M}_i^{\text{NN}}$ are the associated matrix elements.

As the ratio NJET/NN is not known a priori, we must construct criteria on which to reweight. Specifically, we explore the following:

1. A random sample of points (e.g. 10%) regardless of where they are in phase-space;
2. A priori stating which regions of phase-space in which to reweight and then doing so either randomly or over the entire region;
3. Using the NN uncertainties to inform reweighting, e.g. points with large uncertainties are reweighted.

There are several factors informing which approach is the most appropriate. The first of these is the added compute time required: all of these techniques necessitate calculation of the matrix element by an analytic or numerical evaluator and therefore limit the desirable number of points to reweight. The second is the performance gain and confidence in the output in certain regions of phase-space. If the analysis being performed is specific to an under-sampled region of phase-space, such as distribution tails where the network may under-perform due to divergent structures in the matrix element, this could be an especially important region in which to reweight. However, if general process explorations are being performed, meaning all distributions and cross-sections are of relative equal importance, then a less restrictive reweighting on regions of phase-space may be optimal.

In this Chapter, we explore the application of reweighting to the processes described in Section 6.4. While reweighting is not always found to be necessary given the performance of our methodology, we demonstrate how it can be applied and discuss which reweighting criteria show the greatest performance gain.

## 6.4 Results

In this Section, we present the results of our experiments for the $2 \to 3$ and $2 \to 4$ gluon-initiated diphoton amplitudes. As the former is significantly less computationally expensive, we use this for a deep analysis and exploration. The proposed pipeline for using our ML set up and interface with event generators is as follows:

1. Generate an integration grid;
2. Use this with a matrix element provider to generate training and validation datasets;
3. Train the model;
4. Use the model to estimate the values of the remaining phase-space points for event generation while using the same integration grid;
5. Reweight (if necessary);
6. Obtain final results.

To assess performance, we also evaluated matrix elements with NJet in parallel with the models, with different random seeds.

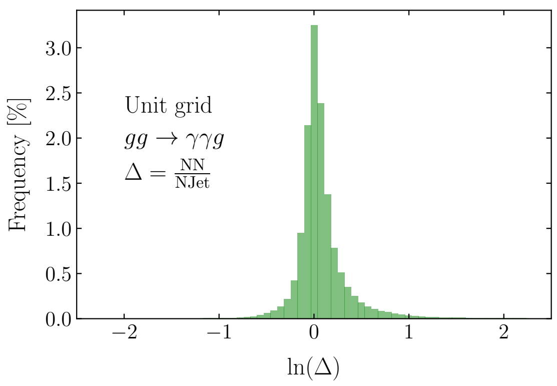


Figure 6.2: NN/NJET errors for the $2 \to 3$ scattering process using a unit integration grid.

### 6.4.1 $gg \to \gamma\gamma g$

First we investigate the performance of our methodology on the loop-induced $gg \to \gamma\gamma g$ process. Following the procedure outlined above, we use a unit integration grid, choose a random seed with which to generate the training and validation datasets, and use another to seed infer on the trained model.

Figure 6.2 shows the performance of our trained NN ensemble at the matrix element level, here represented as the ratio of the model inferred values to the NJET evaluations. The errors form a narrow and approximately symmetric unit-centered distribution, thus demonstrating that the ensemble method has a reasonable per-point accuracy. The slightly elongated right tale of the distribution is due to large matrix element values in highly divergent regions of phase-space, yet these points are in the minority.

Once the ensemble is trained, it is converted to be called by the event generator interface which allows for the calculation of the cross-section and differential distributions. While in Chapter 5 we used a simplified version of the partonic cross-section, here we compute the full hadronic cross-section (see Equation 3.1.1). The first line of Table 6.1 shows the results of the cross-section derived using NJET and the NN ensemble. We see that these two approaches are in excellent agreement, with the ensemble result overlapping within one standard deviation of that calculated by NJET. The errors on the NJET values are the Monte Carlo errors, and the errors

| Cuts | NJET [pb] | NN ensemble [pb] |
|---|---|---|
| Baseline | $4.149 \times 10^{-6} \pm 6 \times 10^{-9}$ | $4.19 \times 10^{-6} \pm 7 \times 10^{-8}$ |
| Baseline + $p_{T,\gamma} > 50\,\text{GeV}$ | $5.283 \times 10^{-7} \pm 8 \times 10^{-10}$ | $5.4 \times 10^{-7} \pm 2 \times 10^{-8}$ |
| Baseline + $m_{\gamma,\gamma} > 50\,\text{GeV}$ | $3.300 \times 10^{-6} \pm 5 \times 10^{-9}$ | $3.34 \times 10^{-6} \pm 5 \times 10^{-8}$ |

Table 6.1: Cross-sectional comparison between NJET and the NN ensemble approach using different cuts. Baseline cuts are those specified at the beginning of Section 6.4. The NJET results are quoted with Monte Carlo errors and the NN ensemble results with precision/optimality uncertainties calculated as described in Section 5.2.3.

on the ensemble are precision/optimality uncertainties. The latter are calculated by training multiple ensembles with different random seeds in the weight initialisation, and in the shuffling of the training and validation datasets. Monte Carlo errors are quoted to one standard deviation and the precision/optimality uncertainties to one standard error on the mean. A more in depth description of this uncertainty analysis can be found in Section 5.2.3.

The error plot and cross-section calculation provide good evidence for the performance of the NN ensemble method both in its ability to learn the distribution of phase-space points on average, as well as its robustness to being integrated into a wider event generation framework with additional phase-space and PDF weights.

Figure 6.3 demonstrates the performance of the NN ensemble in comparison to NJET in six differential slices of phase-space. These include $p_T$, angular, and diphoton system distributions which have been chosen to give a range of realistic constructions exploring different regions of phase-space. In general, the NN ensemble is found to be in good agreement, particularly around the peaks, with the majority of the NN bin values being within the NJET Monte Carlo error. The normalised NN uncertainty on the differential bins is negligible in comparison to the MC error. Strong performance is pronounced in the pseudorapidity distribution, which shows variation at the percent level. The $p_T$ and angular distributions show more fluctuations in the tail events, with the diphoton mass demonstrating the greatest deviations in these regions. However, despite these differences, fluctuations are clearly statistical rather than

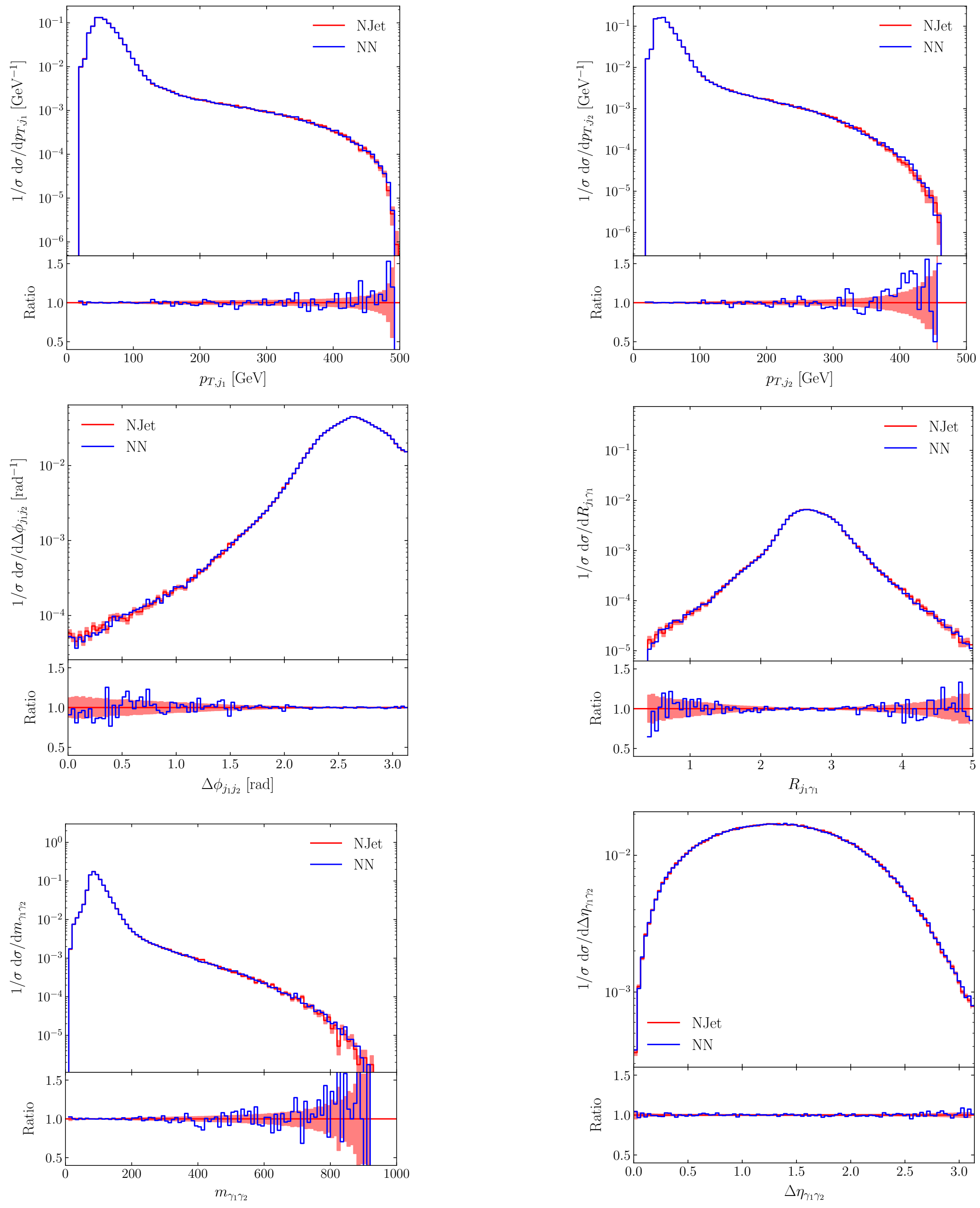


Figure 6.3: Differential distributions normalised to the cross-section for the $2 \to 3$ process comparing NJET (red) with the NN ensemble (blue). The NJET results are quoted with Monte Carlo errors and the NN results with precision/optimality uncertainties calculated as described in Section 5.2.3 but which are negligible in comparison. Pseudojets $j_i$ and photons $\gamma_i$ are ordered by energy, $\Delta\phi$ is azimuthal separation, $R$-separation is defined in Section 6.3.2, and $m_{\gamma_1,\gamma_2}$ and $\Delta\eta_{\gamma_1,\gamma_2}$ are the mass and pseudorapidity separation of the diphoton system.

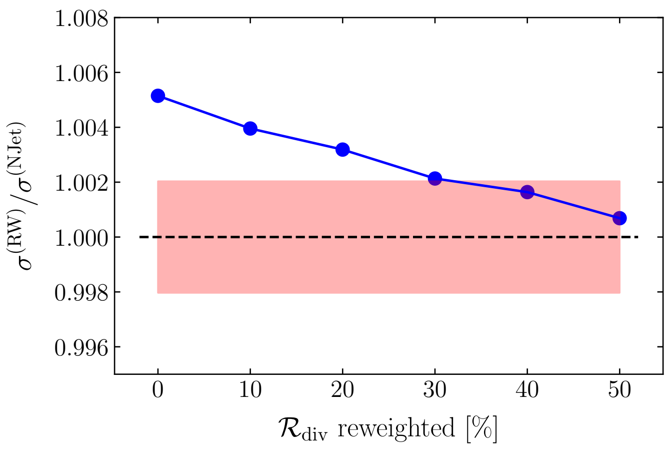


Figure 6.4: Effect of reweighting points in the divergent region of phase-space, $\mathcal{R}_{\mathrm{div}}$, on the ratio between the reweighted cross-section, $\sigma^{(\mathrm{RW})}$, and the cross-section calculated using NJET $\sigma^{(\mathrm{NJET})}$ for the $2 \to 3$ process. In this case, the divergent region comprises approximately 7–9% of the total phase-space (see Appendix E for details). The red band shows the Monte Carlo error on the NJET result.

systematic meaning agreement will increase as the bins are aggregated. This is to be expected given the strong cross-section performance.

The results presented so far have been derived from a NN ensemble trained and tested on the same integration grid and on the same cut parameters. However, in phenomenological explorations it is common to study a range of cut parameters, especially when measuring the effects of new phenomena. Since the NN ensemble performs well at the per-point level (as shown in Figure 6.2), it should also be able to generalise to different cut parameter configurations. Specifically, the ensemble should still be applicable to harsher cuts than those used in training because the it expects the training and testing datasets to be drawn from the same statistical distributions. However, in the event that cuts are relaxed in comparison to those the model was trained on, reweighting could be employed for the relevant additional subset of points, thereby guaranteeing the expected values in these 'unseen' regions of phase-space.

The second and third lines of Table 6.1 present a comparison of cross-section values calculated using NJET and the NN ensemble with harsher cut values than the baseline.

The agreement between the two approaches is comparable to the agreement found before the additional cuts were added, thereby suggesting good generalisability to looser cuts. Indeed, this is not surprising since the points with the largest errors between the NN and NJET were the most divergent points and therefore the ones more likely to be cut, given the IR singularities present in these processes.

The generalisation to additional cut parameters demonstrates both the robustness of this training regime, as well as the practical gain in not having to retrain a network for each specified set of cuts. This allows us to generalise the training and testing procedure outlined at the beginning of this section to suggest that the NN ensemble be first trained on more relaxed cuts and then, as iterations of harsher cut parameters are explored during analysis, these can be applied without the ensemble significantly decreasing in performance. If cuts are to be relaxed then reweighting could be used to ensure good performance at the expense of compute time.

While the network performance has been shown to be strong overall, other reweighing methods can still be explored. Reweighting randomly across all phase-space, even at the 20–40% level, was not found to significantly reduce the difference in the computed cross-sections. Similarly, the NN ensemble uncertainties were not found to be correlated with the uncertainties, and so were discarded as a good reweighting criterion. As mentioned above, the points in which targeted reweighting can be most beneficial are those which fall within the divergent regions of phase-space. Figure 6.4 presents the results of reweighting points randomly in $\mathcal{R}_{\text{div}}$ (as defined in Equation 6.3.1), and shows an improvement in the cross-section when reweighting a greater number of points. This enables the reweighted cross-section, $\sigma^{(\text{RW})}$, to converge to the value calculated by NJET, $\sigma^{(\text{NJET})}$. Indeed, to achieve almost equal values in the cross-sections, the total proportion of phase-space requiring reweighting is at the percent level. Therefore, we find reweighting in the $\mathcal{R}_{\text{div}}$ region of phase-space, and/or when relaxing cuts in relation to those used during training, can improve model performance.

Finally, although the cross-section and differential distributions provide a means to

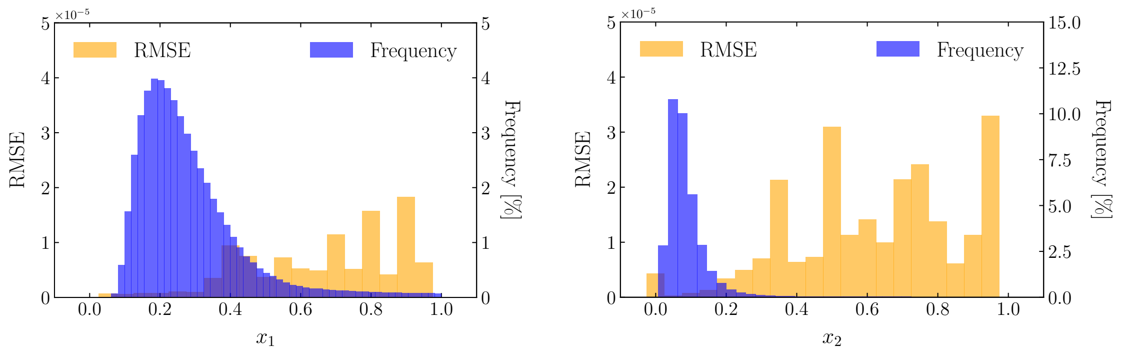


Figure 6.5: Root mean squared error (RMSE) of the NN ensemble approach in comparison to NJet as a function of $x_1$ and $x_2$, and the frequency of points with these values in the training dataset. Frequency differences in $x_1$ and $x_2$ are due to Sherpa sampling differences.

test the robustness of our approach against the additional weights introduced during event generation, we can more explicitly single out the effects of the PDF weights by calculating the NN ensemble error as a function of the momentum fractions, $x_1$ and $x_2$, of the initial state partons. Figure 6.5 shows the root mean squared error (RMSE) of the ensemble as a function of these variables, along with the frequency of points as they appear in the training dataset. As expected, the ensemble performs better in locations with more points, and we see the RMSE grow more significantly in the regions of low-statistics. These regions of strong performance also correlate with large contributions from the gluon PDF, which falls off as $x$ approaches one, and peaks in the low $x$ region. This means that the points most enhanced by the PDFs are those well approximated by the model, whereas those points on which the model performs less well are suppressed. This helps explain why the error distribution in Section 6.4.2 does not look as promising as other examples, yet the cross-section and differential distributions are still well approximated.

### Aside: VEGAS grid optimisation

The results presented so far have used a unit integration grid and RAMBO integrator in order to be process agnostic in the phase-space sampling. As mentioned in Section 6.3.3, however, it is common to use importance sampling and other optimisation

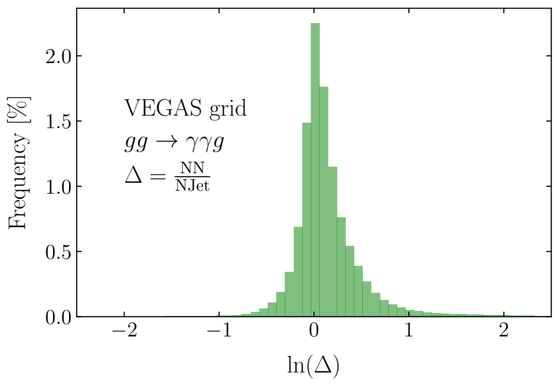


Figure 6.6: NN/NJET errors for the $2 \to 3$ scattering process using a VEGAS optimised integration grid.

techniques to speed up integration convergence. To test the robustness of our approach to these alternative integrators, we use VEGAS during the optimisation grid generation stage. Figure 6.6 shows the error plots for the $2 \to 3$ scattering process using this optimisation setup, while keeping all other setup parameters fixed. Here, we see that the shape exhibited in the error plots is similar to that of the unit grid shown in Figure 6.2, although slightly broader around the peak. This is likely due to the the larger number of points placed in the divergent regions by the VEGAS integrator. The cross-section was also found to be in excellent agreement, with NJET giving $4.151 \times 10^{-6} \pm 1.1 \times 10^{-8}$ pb, and the ensemble giving $4.22 \times 10^{-6} \pm 8 \times 10^{-8}$ pb.

### 6.4.2 $gg \to \gamma\gamma gg$

We now turn to investigate the $gg \to \gamma\gamma gg$ process. Analytic expressions for this process are not available and the numerical implementation is significantly more computationally expensive than for the equivalent $2 \to 3$ process (see Section 6.4.3). Integration grid optimisation is therefore highly inefficient, and so for the remainder of this section a unit grid will be used. To test generalisability, the NN setup is as in Section 6.4.1, with the only change being in the chosen value of $y_p = 0.001$. At higher multiplicity, a greater proportion of points fall within the divergent region, $\mathcal{R}_{\text{div}}$; however, this can hinder model performance by unbalancing the training regime.

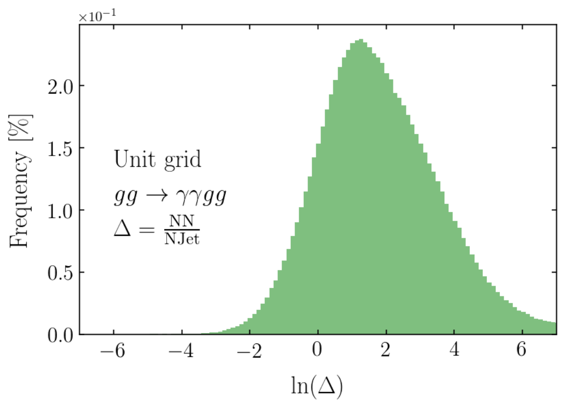


Figure 6.7: NN/NJET errors for the $2 \to 4$ scattering process using a unit integration grid.

It is therefore reasonable to aim to keep the proportion of points in this region approximately constant throughout our experiments, which is achieved by lowering the value of $y_p$ (see Appendix E for more details).

Figure 6.7 shows the performance of our trained NN ensemble at the matrix element level. As expected, the performance has decreased relative to the $2 \to 3$ process shown in Figure 6.2, yet the error distribution is still found to be approximately Gaussian, although with a shifted mean. Despite this, the cross-section calculated using the NN ensemble — $4.5 \times 10^{-6} \pm 6 \times 10^{-7}$ pb — is found to be in excellent agreement with that derived from NJET — $4.9 \times 10^{-6} \pm 5 \times 10^{-7}$ pb. This suggests that although there are several points where the ensemble approach performs poorly, particularly in comparison to the $2 \to 3$ process, these are largely in the PDF suppressed regions of phase-space and found not to affect the cross-section calculation too greatly.

Figure 6.8 shows the performance of the ensemble approach in six differential slices of phase-space. As in the previous example, the ensemble is found to perform well relative to NJET: while noise in the tails of the distributions is still observed, these appear to be reduced in comparison to the $2 \to 3$ process. This further supports the assertion that the points where the ensemble performs poorly are suppressed.

Given the difference in cross-section values calculated using NJET and the ensemble approach, we perform reweighting in the divergent region as discussed in Section

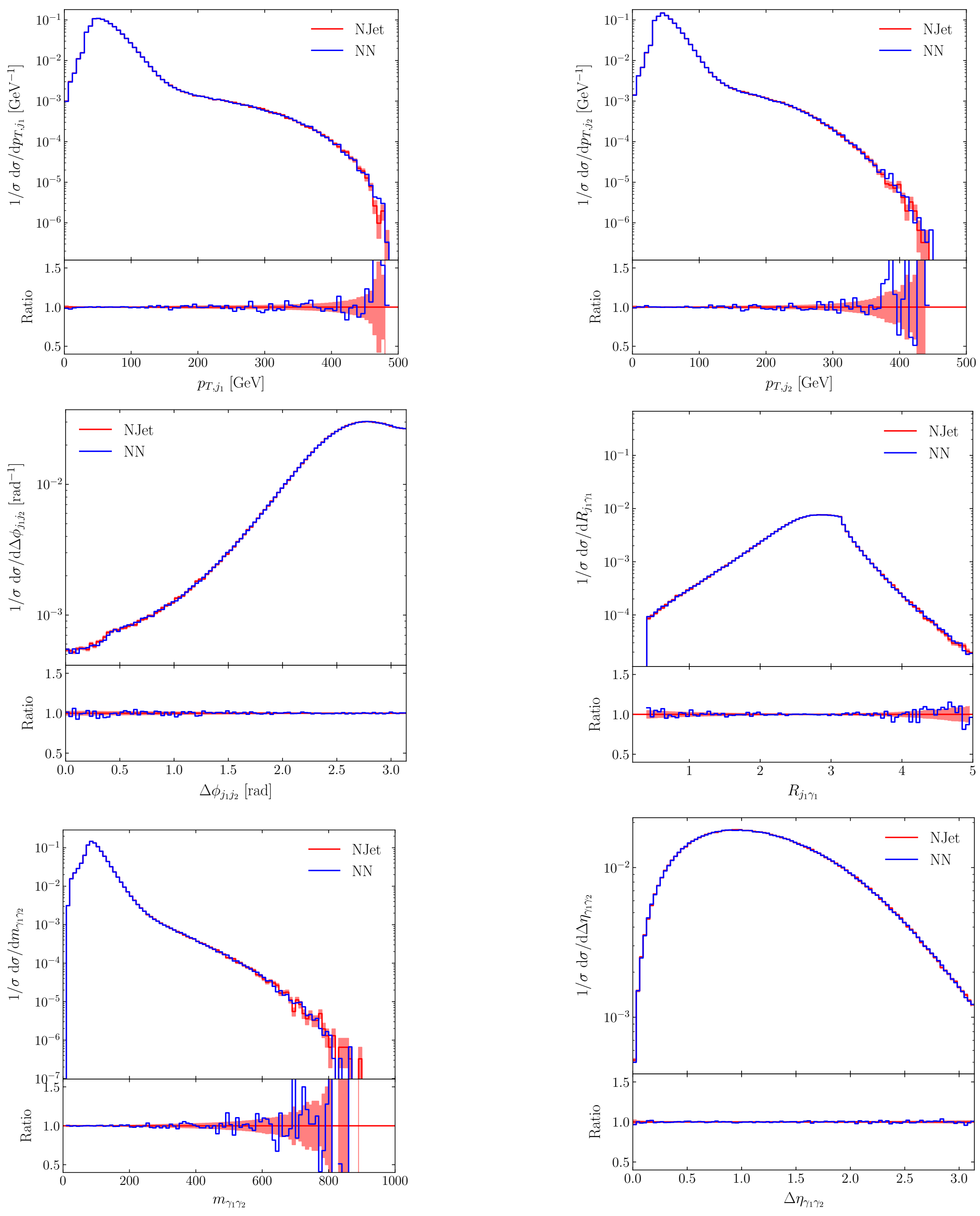


Figure 6.8: Differential distributions normalised to the cross-section for the $2 \to 4$ process comparing NJet (red) with the NN ensemble (blue). The NJet results are quoted with Monte Carlo errors and the NN results with precision/optimality uncertainties calculated as described in Section 5.2.3 but which are negligible in comparison.

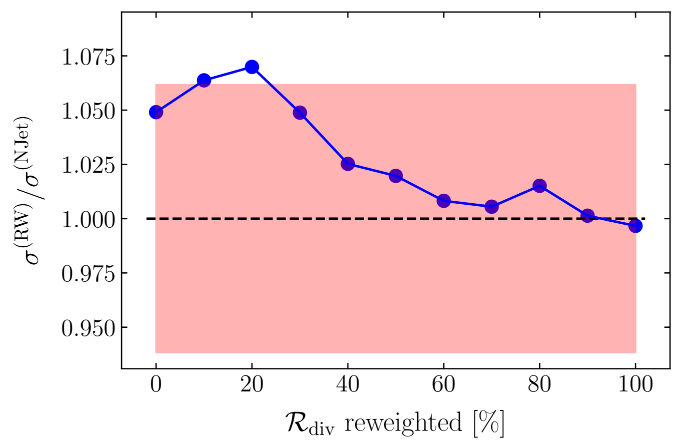


Figure 6.9: Effect of reweighting points in the divergent region of phase-space, $\mathcal{R}_{\mathrm{div}}$, on the ratio between the reweighted cross-section, $\sigma^{(\mathrm{RW})}$, and the cross-section calculated using NJET $\sigma^{(\mathrm{NJET})}$ for the $2 \to 4$ process. In this case, the divergent region comprises approximately 2–3% of the total phase-space (see Appendix E for details). The red band shows the Monte Carlo error on the NJET result.

6.3.4 and Section 6.4.1. As shown in Figure 6.9, reweighting in this region can bring the NN ensemble derived cross-section closer to the value calculated using NJET. In the case of the $2 \to 4$ process, the Monte Carlo error on the NJET result is significantly larger for the same number of points compared to the $2 \to 3$ process. Given these larger errors, and that the ratio $\sigma^{(\mathrm{RW})}/\sigma^{(\mathrm{NJET})}$ resides within these errors, the result is predictably noisy yet still converges, showing that this approach to reweighting can be generalised across multiple processes.

## 6.4.3 Timing

We repeat the performance evaluation of Figure 6.1 with methods involving error estimation as these are likely to be employed in real-world usage. This is comparable with the approach presented in Figure 5.1. For conventional techniques, the dimension scaling test is a standard way to estimate error on the result, and introduces a second matrix element call for each phase-space point evaluation. As discussed in Section 6.4.1, we propose running 20 NN ensembles for each point to obtain a mean with standard error.

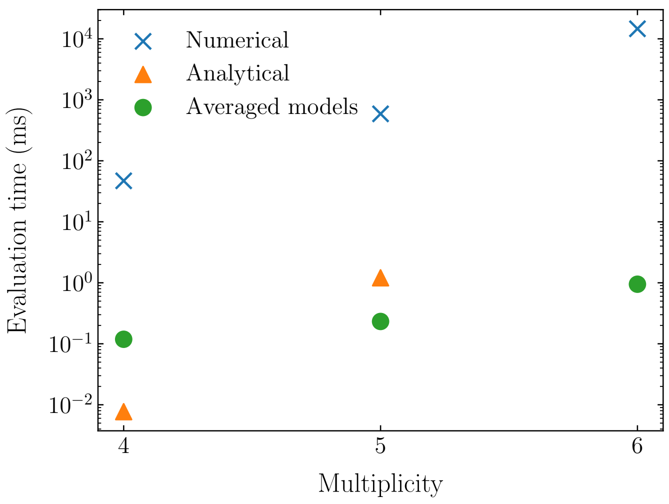


Figure 6.10: Typical per-point call times, for the set of NN ensembles and scaling tests with numerical and analytical techniques, against the number of legs. Compared to Figure 6.1, this incurs a twofold cost on the conventional methods and multiplies the single NN ensemble time by 20. Analytical methods are fastest at $2 \to 2$ and NNs do not offer a dramatic improvement at $2 \to 3$ either, but their fast call time and weak dependence on the number of variables (which scales with multiplicity) win out at high multiplicity. At $2 \to 4$, where no analytical expression is available and extrapolation suggests it would be comparable in call time to numerics, our ML approach is four orders of magnitude faster than the numeric call.

The results, shown in Figure 6.10, demonstrate the per-point speedup in using amplitude NNs in practice. For the $2 \to 4$ process, where amplitude calls dominate conventional simulation time, a $\mathcal{O}(10^4)$ times speedup in amplitude calls is observed which renders the inference stage as negligible in the total time of our NN-based simulation pipeline. Indeed, in comparison to the numerical calculation of the matrix element, the training time of the NN ensemble can also be considered negligible, meaning the total speed up in the overall simulation time is of the order $N_{\text{infer}}/N_{\text{train}}$ — the ratio of the number of inference points to the number of points in the training dataset. This is in agreement with the findings discussed in Chapter 5.

## 6.5 Summary

In this Chapter we provided further evidence that NNs can provide a general framework for the optimisation of high multiplicity observables at hadron colliders. We extended the preliminary studies shown in Chapter 5, for $e^+e^-$ scattering, to hadron-hadron collisions and provided a general interface to the Sherpa Monte Carlo event generator for NNs trained with the NJet amplitude library. In addition, we presented a systematised pipeline for carrying out data generation and training of these NNs. Here, we focussed on the loop-induced processes $gg \to \gamma\gamma + n(g)$ which cause problems for conventional phase-space generation methods and require the computation of expensive scattering amplitudes.

We saw, especially for the $2 \to 4$ process, a good improvement in the total simulation time. Since the calls to the scattering amplitudes dominated the total time, the speed-up was given by the ratio of the number of points used in training to the total number of calls used in the full simulation (during event generation). While this improvement was good to see, it is not the limit of the optimisation. If the trained networks can be used for many subsequent simulations, with different kinematic cuts, the overall improvement would be much greater. Furthermore, we showed that the time-consuming integration grid optimisation stage could be removed meaning the speed-up gained could be considerably more in reality. We showed that our networks reproduce distributions with different cuts in the transverse momentum and diphoton mass without the requirement for retraining, which was very encouraging.

In general, the findings presented in this Chapter are in good agreement with those presented in Chapter 5. Our NN ensemble methodology performed well, producing excellent agreement between the cross-sections calculated by NJet and those derived from the ML model. We also observed a similar growth in NN uncertainties at higher multiplicities, as expected. Overall, the additional validation of our methodology in hadronic collision processes serves as a promising sign for use of this method in future analysis.

# Chapter 7

# Computational Methods for Crisis Response and Epidemic Modelling

Particle physics is a naturally interdisciplinary field, collaborating across the boundaries of many fields including physics, mathematics, and computer science. So far, we have focussed on ML and Monte Carlo-type simulations in the context of particle physics, and more specifically their applications to event generation. These techniques were originally developed in the fields of psychology and nuclear physics respectively, and have since permeated through many other disciplines. In this Chapter we will discuss some of the additional work undertaken alongside that presented in this thesis so far. This work will leverage the techniques and developments in ML and large scale probabilistic computational simulations, and apply them to crisis response and epidemic modelling. This Chapter is not designed to give an in depth view of these projects, but present a high level perspective of the work.

## 7.1 Machine Learning for Crisis Response

During emerging humanitarian crises, such as natural disasters, conflict, or disease spread, data collection to inform planning and relief efforts can be challenging. For example, earthquakes and flooding may make roads inaccessible, meaning response teams have to rely on alternative data sources to inform operations. In these scenarios, machine learning methods can be used to provide timely and large scale data analysis. This Section will focus on applications of ML to satellite image analysis, and the development of ML techniques for tackling the COVID-19 pandemic a various scales.

### 7.1.1 Satellite Image Analysis

When disaster strikes, satellites can be tasked to capture imagery of the affected area, and image analysis can provide a rapid and detailed understanding of the situation. Use cases include: monitoring population displacement, settlement mapping, damage assessment, fire detection associated with human rights violations, damage to transportation networks, floods assessment or identifying direct impact of earthquakes, volcanoes, cyclones and landslides [190]. Furthermore, as crises progress, imagery can be continually analysed for ongoing monitoring, and provide an early warning system for potential changes.

Satellite image analysis tasks performed in humanitarian settings are highly specialised and time-consuming, and expert analysts are required to ensure a high quality of information is returned to teams on the ground. However, in recent years there has been an increase in the number of humanitarian challenges around the world, putting increased strain on analysis work. To aid these efforts, ML techniques have the potential to speed-up many of the time-consuming elements of the analysis, leaving analysts with more time to focus on more nuanced and in-depth studies.

A common task carried out by image analysts is mapping refugee and internally displaced person (IDP) settlements to assess growth, and estimate the number of shelters for resource planning. In the aftermath of conflict and disasters, such settlements are rapidly formed and continuous monitoring of their development, particularly in its early stages, is vital to ensure enough basic aid, tents, blankets, and food are provided. Despite the need for immediate support, mapping these settlements entails the manual counting of shelters visible in a satellite image, a labour-intensive task for which current computer vision techniques may excel.

An early example of the use of ML for humanitarian operations using satellite imagery is presented in [191]. This work develops an ML model for counting the tents, administrative buildings, and other structures, in images of refugee and IDP settlements. The authors use a Mask R-CNN [192] — a type of CNN — trained on images of existing settlements with hand-drawn labels outlining the structures. The model was found to perform well, with an average precision of 0.75 over all settlements in the test dataset. As expected, performance varied between settlements, likely due to the imbalances in the training dataset relative to various settlement features, such as soil colour, settlement layout and structure appearances.

In these sensitive contexts, automated processes providing vital information in decision making must be carefully validated and tuned to maximal performance since e.g. a false positive may be detrimental to human life. To address this, a human-in-the-loop (HITL) system was developed in which analysts validate the results of the model and make corrections as needed. These corrections are fed back into the model to *fine tune* the model to the specific settlement in question. This results in an *augmented* model. As expected, model augmentation was found to further improve results on the settlements used for fine tuning.

We proposed a similar approach to automating the mapping of floods [9]. Floods are the most frequent natural disaster and can cause major societal and economic disruption alongside significant loss of human life [193]. Flooding is usually caused by rivers or streams overflowing, excessive rain, or ice melting rapidly in mountainous areas. Alternatively, coastal floods can be due to heavy sustained storms or tsunamis causing the sea to surge inland. Once an event occurs, a timely and accurate assessment, followed by a rapid response, is crucial.

Traditional methods of satellite image analysis use Synthetic Aperture Radar (SAR) imagery [194]. SAR images can be taken regardless of cloud cover and time of day,

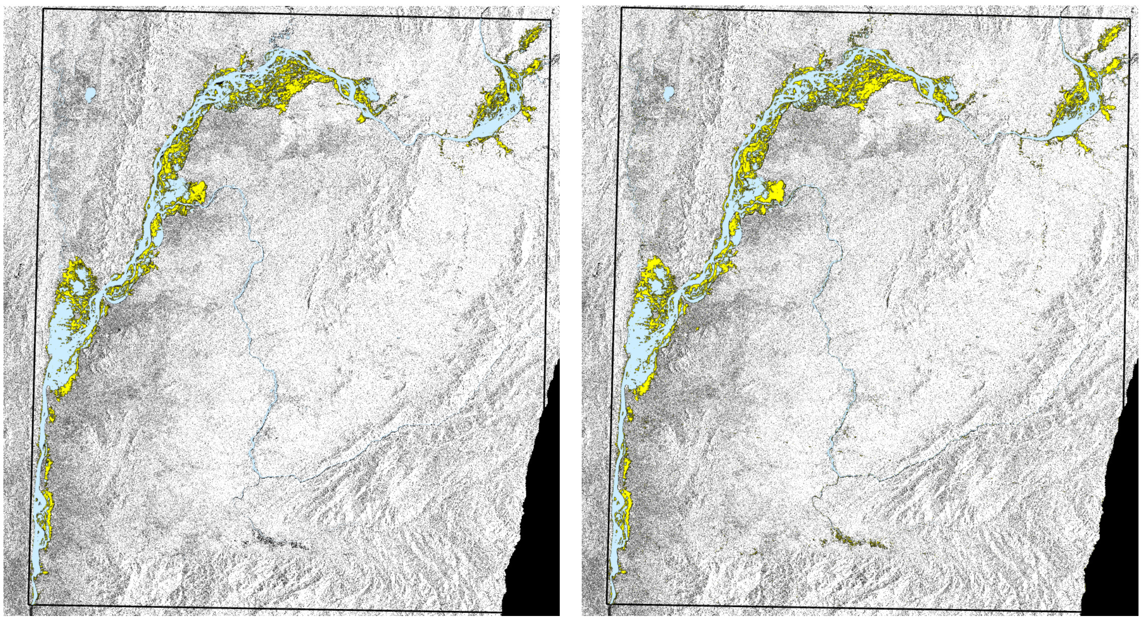


Figure 7.1: Analysis of Saigang Region in July 2019 with permanent water and labels. The images show the area of interest in black outline and the permanent water body derived from the Global Surface Water Dataset [4] in blue. Left: analyst mapped flooded region in yellow. Right: the ML prediction in yellow.

since they use active sensors and specific frequency ranges. Although SAR images generally have a lower resolution in comparison to their high-resolution optical counterparts, high-resolution imagery (at the level of 0.3 - 0.5 m) is not generally required for flood disaster mapping. Due to their popularity, many SAR satellite collaborations, such as COSMO-SkyMed (CSK) [195], TerraSAR-X (TSX) [196], and Sentinel-1 operated by the European Space Agency (ESA), are making for regular and timely image capture.

Using a training dataset of historically mapped floods from different parts of the world, we trained various CNN-type models for semantic (i.e. pixel-wise) segmentation. Given the occurrence of flooding near existing bodies of water, we added existing water bodies back into the historical flood maps (derived from the Global Surface Water Dataset [4]) and the models were trained to detect water in general. We compared the performance of U-NET [197] and XNET [17], with that of a transfer learning approach — this consisted of a RESNET [198] backbone architecture

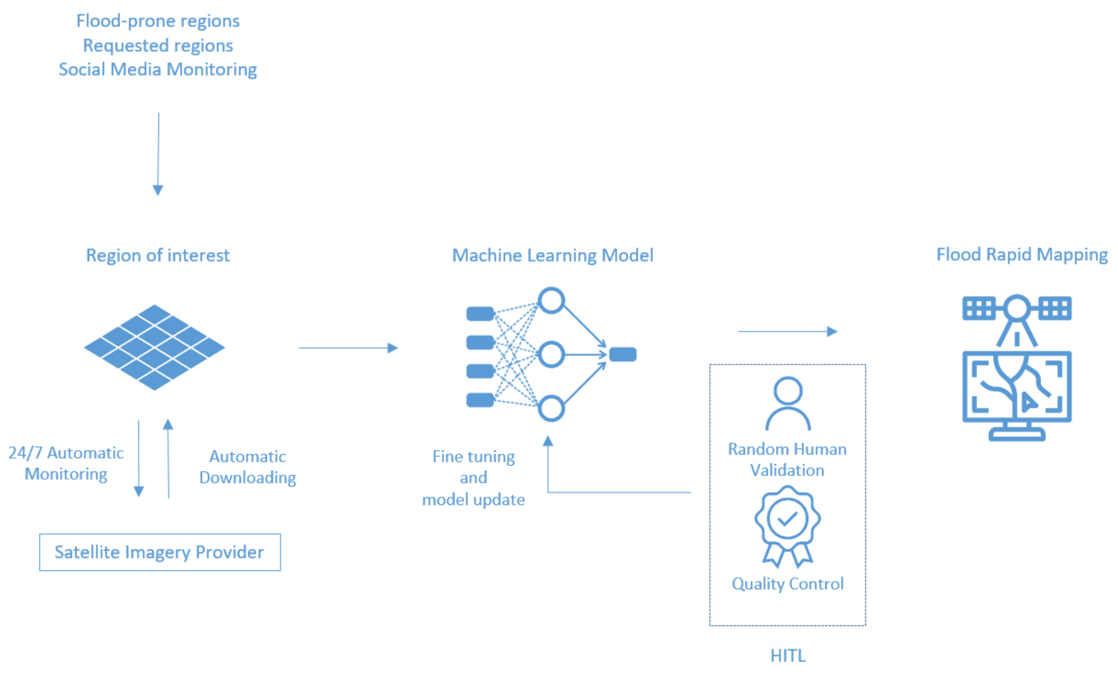


Figure 7.2: Workflow for rapid flood mapping: images can be automatically downloaded from providers based on requests or other activation criteria; the image is fed into a machine learning model for flood detection; human validation and quality control takes place which can also be used to update the model for future floods in that region (resulting in a library of region-specific models); maps are then released.

pre-trained on the ImageNet [199] dataset which replaced the downsampling stage of a U-Net model. The latter approach was found to give the highest accuracy and F1/Dice scores across multiple tests, including testing the ability of the models to detect water and flooded areas in geographical regions with different topographies never appearing in the training dataset. This suggested good generalisation of the approach.

Figure 7.1 shows a comparison between a flood map created by an analyst using classical methods and the output of the chosen ML setup. This image is of a region of Myanmar and no part of this image, or any other images of this region, were contained in the training or validation datasets. This therefore represents an out-of-sample test of the model's performance. Here we see that the model shows good agreement at the high level shown in the Figure. Indeed, the model was found to

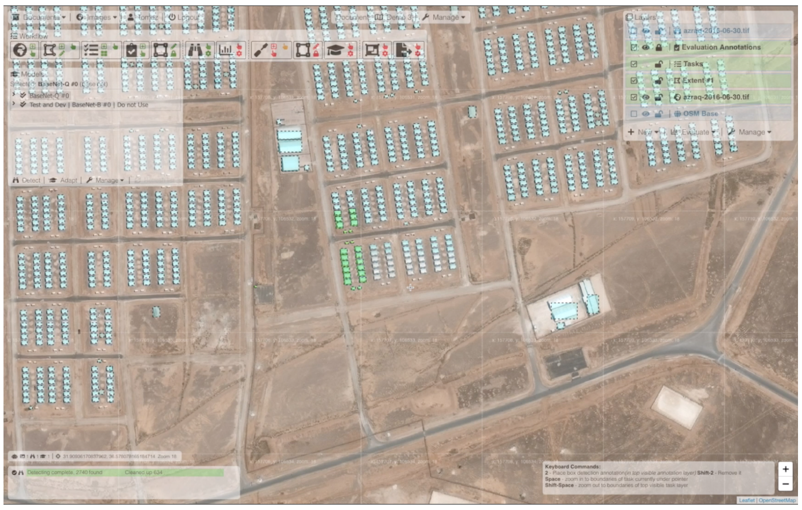


Figure 7.3: PulseSatellite mapping a refugee settlement in Jordan. Buildings detected by the model are highlighted in blue, with analyst-drawn structures in green.

achieve a 99% accuracy and a 0.95 and 0.89 F1/Dice on the water and flooded pixels in the image, respectively.

As in the case of settlement mapping, accurate results are of particular importance in flood response scenarios and so we developed a similar HITL system. A schematic of this approach is given in Figure 7.2, which also shows the additional capability for automatic downloading and processing based on e.g. field requests, or social media monitoring, without the need for manual intervention from mapping analysts. Automating time-consuming stages of the flood analysis pipeline allows analysts to produce maps more rapidly, and therefore generate more timely data. In addition, more regions can be monitored simultaneously and updates can be produced as quickly as images are taken. In total, we found our automated approach to provide $\sim \mathcal{O}(10)$ speed-up over the otherwise manual analysis stages of the pipeline.

Finally, to bring these ideas together, we developed an interactive web-based tool for analysts to interact with the various ML models and enable HITL feedback loops in an automated way — PulseSatellite [10]. Specifically, this platform is designed to host a variety of specialised ML models, fine tuned for different functionalities. Given that each scenario may require different outputs, the tool is dynamic such that new models can be trained in real-time (as users provide feedback on detection

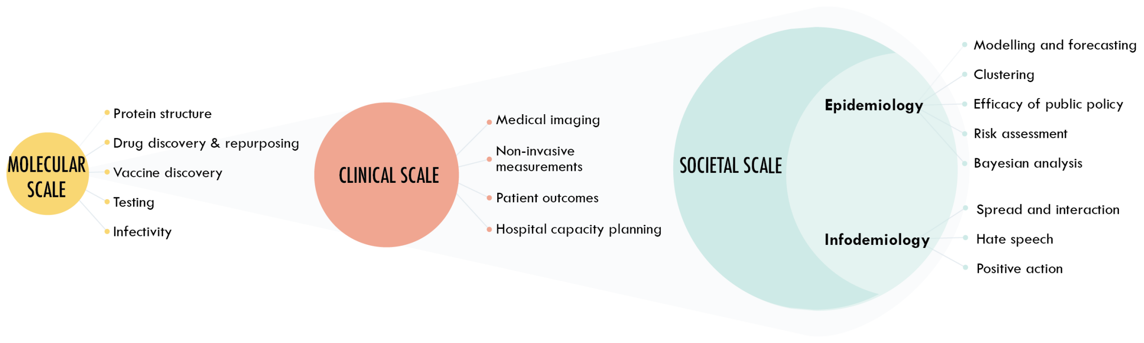


Figure 7.4: AI applications for the COVID-19 response organised at three levels: the molecular scale, the clinical scale, and the societal scale.

results), and is adaptable to new situations. Figure 7.3 shows a screenshot of the tool for the case of refugee settlement mapping described above. The tool was designed alongside humanitarian and satellite image experts to ensure usability by those who are not necessarily familiar with ML software, but who are the experts in rapid crisis response.

### 7.1.2 Mapping the Response to COVID-19

The COVID-19 disease has caused wide-spread fatalities across the world, and researchers have been desperately working to better understand and suppress its spread. Key areas of research include: studying COVID-19 transmission, facilitating its detection, developing possible vaccines and treatments, and understanding the socio-economic impacts of the pandemic. Given the number of unknowns surrounding COVID-19, and the need for a rapid response, numerous approaches utilised ML technology to help discover and tackle the pandemic. Indeed, between January 2020 and August 2020 as many as 50 Artificial Intelligence (AI)/ML papers relating to COVID-19 were being posted on preprint servers such as arXiv, MedRxiv and BioRxiv per week.

In an attempt to provide a resource to the research community, we conducted an exercise to map the landscape of AI applications being used to fight the pandemic [11]. We categorised applications at three levels: *molecular*, where AI was being used to identify new or existing drugs for treatment; *clinical*, in which AI was supporting diagnosis and evaluating prognosis based on medical imaging and non-invasive measures; and *societal*, where AI was tracking both the epidemic and the accompanying infodemic using multiple data sources. We also reviewed datasets, tools, and resources needed to facilitate AI research.

In a review of works up to August 2020, we found that very few systems had achieved operational maturity. However, there were several promising results including the use of ML for analysing CT and X-Ray scans for early COVID-19 detection, as well as several efforts to monitor and fight the *infodemic* of online misinformation in online social media. We discussed these promising directions, as well as some of the associated risks and pitfalls in more detail in follow up work [13]. Despite these efforts, for AI applications addressing COVID-19 to have a global impact large-scale data and model sharing, operational validation, and adaptation to local contexts are needed. This requires cooperation and solidarity across borders as well as the involvement of many relevant parties, including healthcare workers.

Through our review, we identified three key calls to action. Firstly, we believe that scalable approaches to data and model sharing using open repositories will drastically accelerate the development of new models and unlock data for the public interest. Global repositories with anonymised clinical data, including medical imaging and patient histories, can be of particular interest in order to generate and transfer knowledge between medical institutions. To facilitate the sharing of such data, clinical protocols and data sharing architectures will need to be designed and data governance frameworks will need to be put in place. However, we must ensure that biases are addressed — currently much of the data collected and released in existing datasets is from Global North countries, leaving others underrepresented. This can have serious implications if such data is used for training models which are then deployed in other regions [14].

Second, the multidisciplinary nature of the research required to deploy AI systems in this context calls for the creation of extremely diverse, complementary teams and long-term partnerships. Funding opportunities which encourage such collaborations and define key research directions may help accelerate the success of such partnerships.

Finally, we believe that open science and international cooperation can play an important role in this pandemic that knows no borders [12]. Proven solutions can be shared globally and adapted to other contexts and situations, prioritising those solutions that target local unmet needs. In particular, given that many international organisations, private sector companies, and AI partnerships operate across international borders, they may be in a position to facilitate the knowledge dissemination and capacity building of national health systems. Regions with less capacity can benefit from global cooperation, as well as concentrating their efforts on the most important local challenges. AI systems, methods, and models can act as a compact form of knowledge sharing which can be used in, and adapted to, other contexts if they are designed to be widely deployable, requiring low energy and little computing resources.

## 7.2 Epidemic Modelling

In Section 7.1.2 we discussed the applications of AI to help stop the spread of COVID-19. Over the course of the pandemic, national and local governments have implemented numerous interventions, such as reductions in movement, closure of certain locations, as well as complete ‘lockdowns’. The assessment of such policy options to mitigate the impact of this and other epidemics on the health of individuals and the efficiency of healthcare systems, relies on a detailed understanding of the spread of the disease, and requires both short-term operational forecasts and longer-term strategic resource planning. Epidemic modelling can provide an important set of data to inform such assessment and decision making.

There are various modelling approaches which aim to provide insights into the spread of an epidemic. They range from analytic models, formulated through differential or difference equations, which reduce numerous aspects of the society–virus–disease interaction onto a small set of parameters, to purely data-driven parameterisations, often based on ML, which inherently rely on a probability density that has been fitted to the current and past state of the system in an often untraceable way. Indeed several such approaches were included in the review discussed in Section 7.1.2.

Another class of approaches, agent-based models (ABMs), are "*particularly useful when it is necessary to model the disease system in a spatially-explicit fashion or when host behaviour is complex[.]*" [200]. [1] Being the traditional tool of choice to analyse behavioural patterns in society, they find ample use in understanding and modelling the observed spread of infections and in leveraging this for intermediate and long-term forecasting [202–204]. Such models also provide the flexibility to experiment with different policies and practices, founded in realistic changes to the model structure, such as the inclusion of new treatments, changes in social behaviour, and restrictions on movement.

### 7.2.1 The June Framework

To simulate epidemic outbreaks, specific realisations of ABMs, individual-based models (IBMs), have been developed in the past two decades, for example [205, 206]. In these models, the agents represent individuals constituting a population, usually distributed spatially according to the population density and with the demographics — age and sex — taken from census data. Evidence from disease data such as COVID–19 fatality statistics suggests that case and infection fatality rates are correlated, amongst other factors, to the age and socio-economics status of the population exposed to the etiological agent [207]. This necessitates the construction of a model with exceptional social and geographic granularity to exploit highly local heterogeneities in the demographic structure.

To address this challenge, we developed the June framework [15] — a generalisable

[1] Indeed, many models also feature some optimising behaviour of individuals as artificial intelligence-type actors against randomly drawn welfare functions, see for example [201].

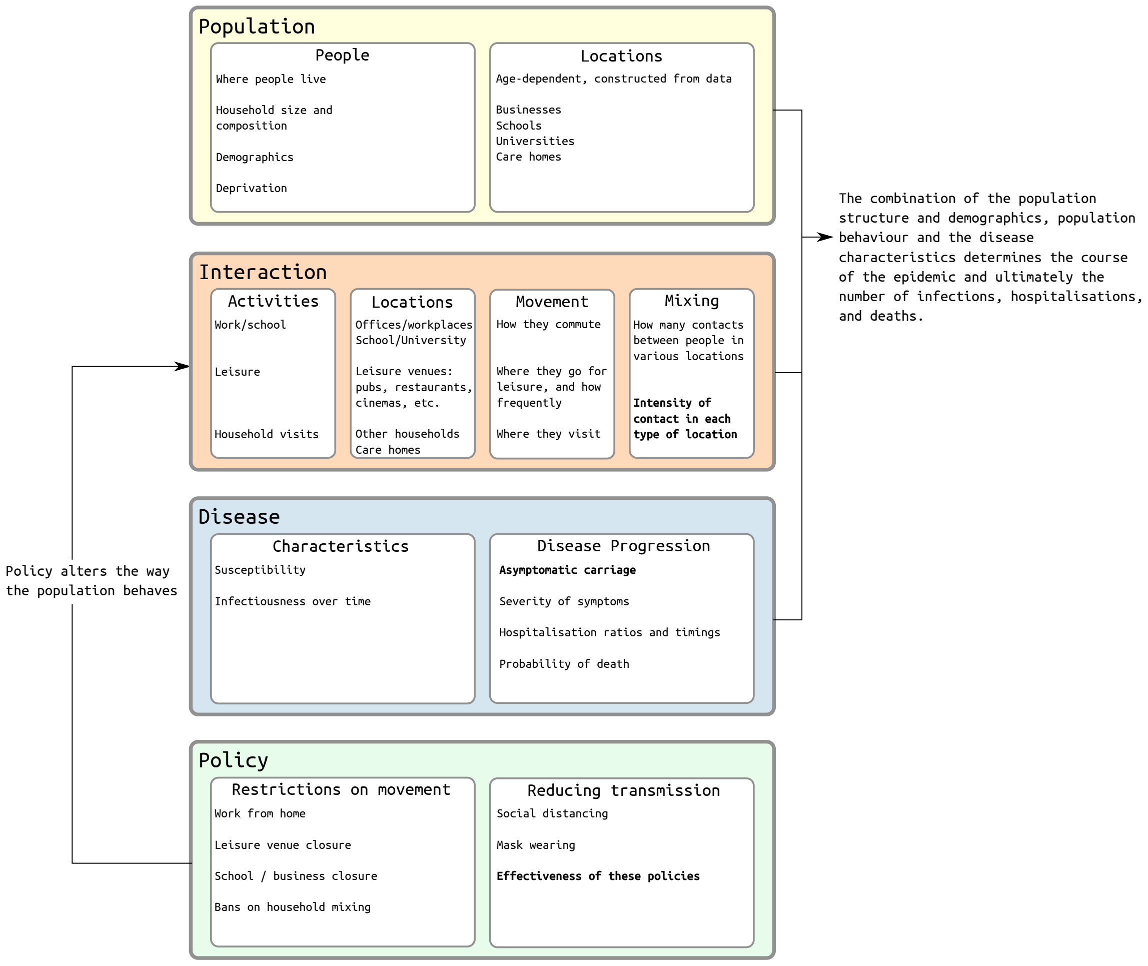

Population
People
Where people live
Household size and composition
Demographics
Deprivation
Locations
Age-dependent, constructed from data
Businesses
Schools
Universities
Care homes
Interaction
Activities
Work/school
Leisure
Household visits
Locations
Offices/workplaces
School/University
Leisure venues: pubs, restaurants, cinemas, etc.
Other households
Care homes
Movement
How they commute
Where they go for leisure, and how frequently
Where they visit
Mixing
How many contacts between people in various locations
**Intensity of contact in each type of location**
Disease
Characteristics
Susceptibility
Infectiousness over time
Disease Progression
**Asymptomatic carriage**
Severity of symptoms
Hospitalisation ratios and timings
Probability of death
Policy
Restrictions on movement
Work from home
Leisure venue closure
School / business closure
Bans on household mixing
Reducing transmission
Social distancing
Mask wearing
**Effectiveness of these policies**
Policy alters the way the population behaves
The combination of the population structure and demographics, population behaviour and the disease characteristics determines the course of the epidemic and ultimately the number of infections, hospitalisations, and deaths.

Figure 7.5: Overview of the structure of June. Free parameters to be fitted or estimated are shown in bold.

modular framework for simulating the spread of infectious diseases with a fine-grained geographic and demographic resolution and a strong focus on the detailed simulation of policy interventions. The framework is built on four interconnected layers: the *population* layer encodes the individuals in the model and constructs static social environments such as the households they live in, the schools they study in, and the workplaces where they work; the *interaction* layer models the social interactions of individuals, based on data about the frequency and intensity of contacts with other people in social settings; the *disease* layer models the characteristics of disease transmission and the effects it has on those infected — whether they are asymptomatic or symptomatic and their trajectory through the healthcare system; and the *policy* layer allows policies such as physical distancing, changes in behaviour, and the closure of certain venues to be modelled at a corresponding granularity. A schematic of the framework is given in Figure 7.5.

June is designed to be highly modular, allowing for adaptation to different geographical regions, daily routines, as well as multiple circulating diseases. Interactions and disease transmission are modelled probabilistically at each time step in the model. Similarly, agents can be given free choices of where to go during certain parts of the day. These choices are made based on a series of Poisson processes which make a randomised decision. Clear parallels can therefore be drawn between a multi-agent simulation of people and a multi-agent simulation of particle interactions — both leverage probabilistic modelling, randomised Monte Carlo-type processes, and require large compute times to rerun simulations to ensure multiple possible interactions and permutations are explored.

### 7.2.2 Modelling COVID-19 in England

To model the spread of COVID-19 in England, we used multiple datasets collected by the Office of National Statistics (ONS), such as census records and household surveys, to initialise the population. Each agent in the population was assigned a set of demographic attributes: age, sex, ethnicity and socio-economic index. ONS data was then used to cluster the population into representative household structures which determined the families of the agents — since household transmission is a key route for disease spread, matching the data at a detailed level is particularly important. Agents who were old enough to work, were assigned work places which they would visit each weekday, and the children were assigned to schools in the model. School classes were broken down by year groups with a higher probability for intra-year mixing than inter-year, to represent realistic interactions.

At certain times of day the agents are given free choices about where to go — e.g. going to a pub or restaurant after work. To assign such activities to agents, we first check if the individual does any activity at the given time step

$$\begin{aligned} p\ (\text{ any activity} \mid \text{age}, \text{sex}\ ) &= 1 - p\ (\text{ no activity} \mid \text{age}, \text{sex}) \\ &= 1 - \exp\left( - \sum_{a=1}^{N} \lambda_a(\text{age, sex})\ \Delta t \right), \end{aligned} \tag{7.2.1}$$

where $\lambda_a$(age,sex) is the Poisson parameter associated with activity $a$ for a person with a given age and sex (determined from ONS surveys, and similar literature), $N$ is the number of possible activities and $\Delta t$ is the amount of time allowed to do a given activity. If no activity is performed then the individual returns to their home.

If a person carries out an activity, the next step is to determine which specific activity is chosen. The probability that activity $a$ is chosen, given that the person does any activity is given by

$$p_a \quad = \quad \frac{\lambda_a(\text{age, sex})}{\sum_{j=1}^{N}\ \lambda_j(\text{age, sex})}. \tag{7.2.2}$$

When any group of agents in the model are in the same location at the same time, they have a chance to interact. If one of them is infected, we compute the probability that they infect another agent based on numerous factors including: contact patterns derived from surveys [208, 209], the infectiousness of the infected agent, the susceptibilities of the uninfected agents, and the intensity of contacts in that given location — these parameters are fitted and account for differences in contact intensity between e.g. family members and people in a restaurant.

Finally, we implemented the majority of policy interventions enacted by the UK Government. Since we model at the individual level, and we know the locations where agents live, work and socialise, we were able to control their movement at a highly granular level. However, since compliance levels with government policy spatio-temporally vary, agents probabilistically decide whether to comply with the policies or not.

The complexity of the model, and the diversity of locations where agents can interact, leaves us with 18 free parameters for fitting. These largely correspond to the interaction intensity parameters, as well as several policy compliance rates. Due to the high dimensionality of the parameter space, fitting the model is challenging. We employ the Bayes linear emulation and History Matching methodology [210–212], a widely applied uncertainty quantification approach designed to facilitate the exploration of large parameter spaces for expensive to evaluate models of deterministic or stochastic form. We then identify a set of particular model outputs to match to corresponding observed data. Here we focus on hospital deaths (CPNS [5]) and total deaths (ONS) at well-spaced time points throughout the period of the first wave of the epidemic. [1] Bayes linear emulators are then constructed for each of the model outputs at each of the chosen time points and iterative History Matching [211, 212] is used find an optimal set of parameters at which the model fits the data well.

Results of this procedure are found in Figure 7.6. Here, we show how the model is able to simultaneously produce fits to recorded hospital deaths, disaggregated by regions of England. It is important to note that we do not fit a different model to each region. Each model is fitted for the whole of England while managing to simultaneously capture regional differences in death rates. Similar results are found at the age stratified level. Given the level of granularity of the June framework,

[1]These datasets were used for fitting since they are likely to be more accurate in comparison to other sources such as case numbers. To account for various known discrepancies, such as inconsistent weekend reporting, corrections and smoothings were applied. Seroprevalence studies conducted at single time points were used to validate the fits.

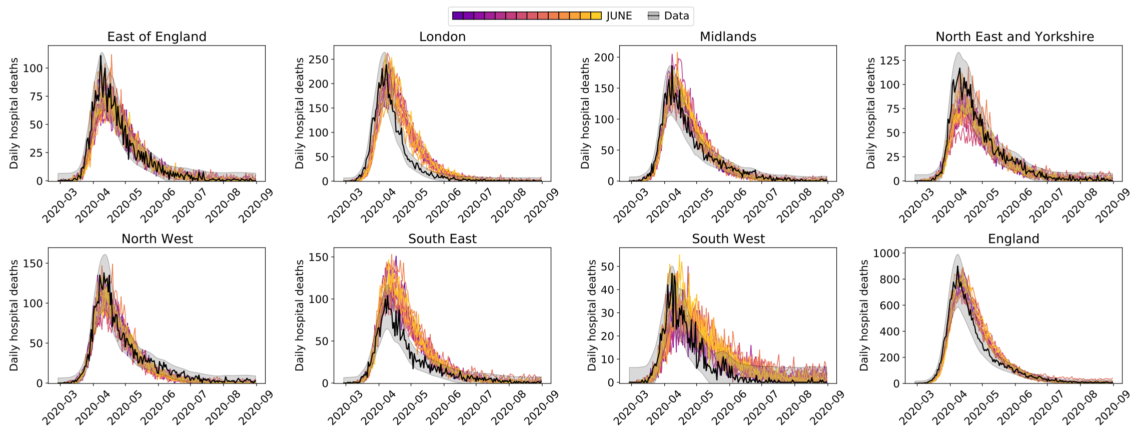


Figure 7.6: Daily hospital deaths for each region in England, and England overall, for 14 realisations of JUNE as described in this section. Each realisation is illustrated as a separate colour for visibility. Observed data in black with 3 standard deviation error bands. Data from CPNS [5].

and the ability to fit such a complex model, results were used to inform planning by the National Health Service (NHS) and Public Health England — examples include predicting the onset of the second wave experienced in September which allowed decision makers to plan for an earlier wave onset than previously thought, as well as the ability to probe the model to help understand why certain regions were experiencing case surges, while others were not. As COVID-19 continued to spread throughout the population, adjustments and additions were made to the model to account for new policies, variants of concern, and vaccination strategies. Recent analysis has found that our model is able to reproduce social and ethnic inequalities observed in seroprevalence studies conduced in July 2020. Work is ongoing to understand how and why JUNE reproduces these without directly using these characteristics in any probabilistic modelling.

### 7.2.3 Modelling COVID-19 in Refugee and IDP Settlements

The spread of infectious diseases such as COVID-19 presents many challenges to healthcare systems and infrastructures across the world, exacerbating inequalities

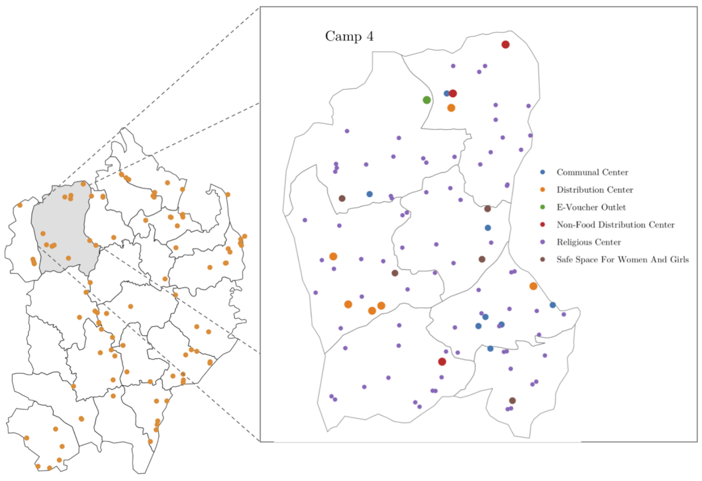


Figure 7.7: Left: modelled distribution centres. Right: Detailed view of Camp 4 showing six types of modelled locations.

and leaving the world's most vulnerable populations most affected. Given their density and available infrastructure, refugee and internally displaced person (IDP) settlements can be particularly susceptible to disease spread. We presented an adaptation of the June framework to model disease spread in these settlements in [16]. Specifically, we focus on the Cox's Bazar settlement in Banglandesh — the largest settlement in the world, containing approximately 900,000 people.

A similar procedure to that described above was taken to initialise the population in the model. Census data was available from the United Nations Refugee agency (UNHCR), which gave a highly granular breakdown of the age and sex of the population, as well as family structures. Since space is limited in the settlement, many families share a shelter with another family. This was mirrored in the model through the random clustering of families, while ensuring we matched statistics on the average shelter size.

Unlike the majority of the England population, most of the population of the Cox's Bazar settlement do not have a fixed daily routine structure (with the exception of children going to school for 2 hours per day). There are many locations which they might visit in the settlement — see e.g. Figure 7.7 — and the majority of these are open every day. We therefore assign each agent five time steps in each simulated day, during which they choose randomly which activity too attend in accordance with the method outlined in Equation 7.2.1. The Poisson parameters were derived from available literature and surveys of the population in the settlement. The implementation of the disease layer was the same as in the case of England, while adjusting for the different comorbidity prevalences between England and the population in the settlement, which may affect the likelihood of severe symptoms and hospitalisation.

COVID-19 testing and case reporting data in the Cox's Bazar settlement was incomplete, and so interaction intensity parameters could not be fitted as we did in the case of England. We therefore estimated a baseline (no intervention) based on the limited data available and focussed primarily on analysing the efficacy of possible interventions through comparing the relative magnitudes of infection curves between various implementation conditions — a *scenario-based modelling approach.* Different models and approaches can account for different degrees and types of uncertainty, making consensus on statistical predictions challenging even in more data-rich environments. However, despite often highly variable predictions, consensus can often be reached on ranking intervention efficacy [213] which can be of interest for decision making.

To support decision making in the settlement, we simulated numerous intervention strategies based on those deemed most important by public health officials. This consisted of an assessment of short and medium-term needs, as well as feasibility and timeliness of possible interventions. These included: the offering of different home-case treatment mechanisms, the effects of mask wearing based on compliance and mask efficacy, and reopening schools (learning centres). In the latter case, we found that reopening schools could increase the rate of transmission, and that cases would not be confined to the younger ages groups, but quickly broke out into all age strata. As this posed an increased risk to those most vulnerable, we assessed possible strategies to safely reopen the schools. We simulated the effects of children attending their school only every other day, allowing for increased physical

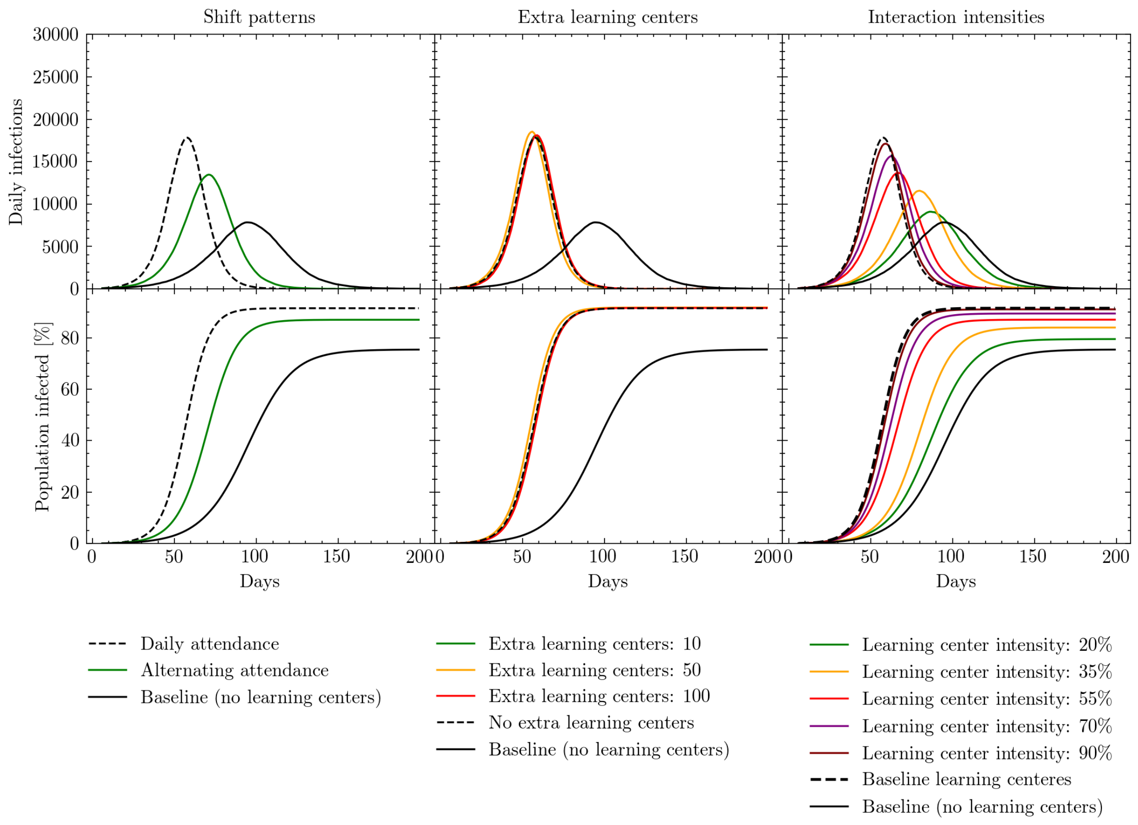


Figure 7.8: Simulated daily (7-day rolling average) and cumulative infections measured in days since the beginning of the simulation. Black solid lines represent the baseline policy in which learning centres (schools) are closed. Black dashed lines represent the policy in which learning centres are open with no additional mitigation strategies.

distancing, the opening of additional schools, and the effects of reducing the intensity of interactions in the schools. Figure 7.8 shows the results of these simulations in which the alternating attendance is found to have an important effect, while reducing the intensity of interactions to 20-35% of their previous value can have a significant impact on daily infection statistics. We found that the upper end of this relative intensity range could correspond to enforcing mask wearing alone if compliance and the efficacy of the masks worn are high, or a combination of mask wearing and physical distancing is implemented. The lower end may correspond to the combination of physical distancing in classes (enabled through alternating attendance), mask wearing and improved ventilation [214–218].

By simulating multiple scenarios we were able to deliver results to decision makers which explored a range of options of varying degrees of feasibility. Our findings from simulating the effects of case isolation in treatment centres, based on how rapidly someone could be quarantined after testing positive for COVID-19, allowed public health officials to draw up contingency plans for the case in which treatment centres were overrun. Similarly, our work on mask wearing simulations informed decision makers that home-made masks using materials available to the population of the settlement could significantly slow disease spread if high enough compliance levels were reached. This meant that resources could be allocated to mask-making training programmes, which gave those in the settlement greater ownership over their protection, as well as communication strategies to promote mask wearing. These modelling efforts provided detailed insights, and the development of a flexible framework for rapid testing of future scenarios.

## 7.3 Summary

In this Chapter we have discussed some of the additional work conducted alongside the research previously discussed in this thesis. This work has utilised many of the techniques used and developed in particle physics, and applied them to the domains of crisis response and epidemic modelling. They have also all been interdisciplinary efforts in themselves, involving collaborators from across the United Nations (UN), non-governmental organisations (NGOs), and other university research institutions. These collaborations were not just important and necessary to bring domain knowledge, but by working with experts who could benefit from these techniques and methodologies, we could ensure the results were useful and actionable.

Much of this research has been focussed on the inclusion of often underrepresented groups and geographical regions. Training models for flood detection on highly disaster-prone regions, such as Bangladesh and Mozambique, means that response in these areas can be better informed and increasingly rapid. The development of a web-based tool for performing such analysis also means that field operatives and emergency responders can have access to the information and analysis themselves, thus better equipping them when disaster strikes. Similarly, the development of a highly granular model for simulating epidemic spreads has allowed us to begin to better understand the origins of ethnic and socio-economic disparities in COVID-19 infection and mortality rates. The extension of this work to understanding epidemic spread in refugee and IDP settlements containing vulnerable communities, generally underserved by the modelling community, has provided experts with more relevant tools for making decisions on public health interventions, and we hope this work inspires similar efforts now and in the future.

Additional work is also ongoing to develop ML approaches, and other statistical methods, for scenario-based modelling to aid in cross-border migration prediction and contingency planning in Latin America. These tools enable powerful multi-variate analysis and, since the pull/push factors for migration flows are highly complex and situation dependent, they present an interest candidate for modelling current and future patterns. Similarly, ML applications to social media analysis are widespread. However, radio is one of the most widely used channels for communication and information dissemination globally, yet analysis methods are minimal. In future work we will present ML methods for analysing transcribed radio stations in multiple countries in sub-Saharan Africa, with the objective of better understanding the discussions and discourse. Such approaches can help inform response efforts to crisis situations, such as monitoring and understanding aspects of infodemics in 'offline' communities.

Through diverse and interdisciplinary research across multiple geographic regions, we hope to better work towards one of the fundamental pillars of sustainable development — leaving no one behind.

# Chapter 8

# Conclusions

This thesis has discussed the potential for using machine learning techniques at multiple parts in the event generation pipeline for fundamental particle interactions. Numerous works have focussed on optimising phase-space sampling for more efficient integration, parton showering, as well as applying machine learning to simulate the full event generation pipeline. However, few works have addressed the computationally expensive task of high multiplicity matrix element calculation.

We have also briefly discussed applications of these methods, which we introduced in the context of particle physics, to other domains. In particular, we demonstrated how ML techniques can be used in response to crises — focussing on understanding mass migration and mapping floods to inform response teams, as well as the emergence of a vast number of approaches directed at challenges presented by the COVID-19 pandemic. Event generation technology is founded on the probabilistic simulation of multi-particle interactions, and we also discussed how similar methods could be applied to epidemic modelling, both at the national level, and to help some of the most vulnerable communities living in refugee and IDP settlements around the world. Interdisciplinary work such as these simultaneously leverages expertise in multiple domains, and the complexity of modern challenges necessitates more interdisciplinary work to overcome them.

Indeed, there is much to be gained by the particle physics community from those in other domains. ML is a rapidly growing fields with new methods constantly being developed. With the challenges to physics calculations discussed in this thesis only growing in the future, as experiments become increasingly precise in their measurements, it is important to remain up-to-date with this research and keep trying new techniques. Similarly, a growing number of researchers are focussing on probabilistic processes and leveraging technological advances in automatic differentiation libraries, and hardware specifically designed for matrix algebra, to make previously highly demanding inference problems tractable. These research directions could have applications, for example, in improving importance sampling methods for efficient integral convergence. Along with those already discussed above, techniques developed in the field of particle physics can also be applied back to these domains — for example, in striving for a better understanding of uncertainty in ML and probabilistic processes, as well as providing a test bed for complex learning tasks. To ensure more shared learning and collaborations, we should be looking to continue to open up our research and conferences, and encourage the creation of spaces where cross-disciplinary discussions can take place and funding is available to allow for such flexibility and potentially unorthodox high risk/high reward research.

The main focus of this thesis has been developing machine learning approaches for matrix element approximation up to and including one-loop calculations at high multiplicity. Our proposed pipeline includes training an ensemble of NNs which divide the scattering amplitudes into IR divergent sectors according to the FKS mapping, and finds excellent agreement between distributions generated with the networks and those generated with conventional analytic and numerical approaches. Errors from the NN were included through variations of training parameters and were found to grow with the complexity of the problem, as anticipated (although they remained largely within the percent level). We also showed that by reweighting the generated events according to their divergence structure, the accuracy of the simulation could be improved at a rather low additional computational cost. This step also provided additional confidence in the inference of the network.

We validated our methodology on both leptonic collision processes at high multiplicity and one-loop level, as well as hadronic collisions focussing on complex loop-induced processes. These are particularly phenomenologically relevant processes given the increasing precision of particle collision experiments, which are reaching percent level measurements, requiring increasingly higher precision calculations in perturbative QCD. High multiplicity matrix elements are time-consuming to calculate using existing technology, and are one of the bottlenecks in event generation simulations. The development of new techniques for calculating these quantities, such as those presented in this thesis, are therefore required. Indeed, this will only become more relevant after LHC upgrades such as HL-LHC.

Overall, our methodology has shown to provide large speed-ups in the computational time required to perform these calculations, meaning time-consuming processes now have the potential to be included in large event generation simulations without incurring dramatic CPU costs. Improvements to the training and inference stages of the NNs could also be provided through GPUs, which may be important if a large number of variations in the cut analysis are required. Indeed, with more research on the use of such hardware for other components of event generation [219–224], integration of GPUs into modern event generation pipelines may become increasingly easier. Futher, the distribution of the trained networks is also simpler than the large quantity of data generated with Root Ntuples [225, 226], another technique for optimising the information that can be extracted from expensive simulations, thereby multiplying the potential computational gains.

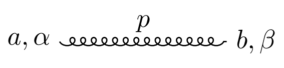
There remain open questions of course. It would also be very interesting to apply this technique to the more intricate problem of real radiation event generation, since NLO and NNLO simulations are often dominated by these contributions. We may also find that effects of higher order, double unresolved singularities begin to play a role. Since NNLO sector decomposition strategies are available, it would be good to explore this direction in the future. A robust framework for estimating ML uncertainties is still not available and several sources of uncertainty are still unaccounted for or hard to quantify, as discussed in Section 5.2.3. Finally, future work to make connections between the amplitude-level approach presented here and those ML approaches focusing on other parts of event generation, and to assess the compounding of the ML-induced uncertainties, would be beneficial. For example, linking together some of the advances in phase-space sampling and integration with our approach, and propagating the uncertainties, could start the development of an integrated ML pipeline for event generation. We hope that these studies will help to develop a benchmark for future work and lead towards a general framework that can be used in future experimental analysis.

# Appendix A

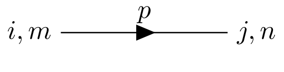
# QCD Feynman Rules

Feynman rules can be derived from the Lagrangian of a theory and define the permitted interactions in the theory, and how to compute the matrix elements of certain allowed processes. In this Appendix we present the Feynman rules for QCD, derived from the Lagrangian in Equation 2.1.1. These rules match the form of those given in [28]. [2]

In massless QCD (which is the focus of this thesis), the propagators of gluons and quarks with momentum $p$ are given respectively in the Feynman gauge by



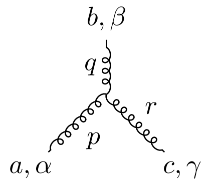
$$= \delta^{ab}\frac{-ig^{\alpha\beta}}{p^2+i\epsilon}, \tag{A.0.1}$$



$$= \delta^{ij}\left.\frac{i}{\not{p}+i\epsilon}\right|_{nm} = \delta^{ij}\left.\frac{i\not{p}}{p^2+i\epsilon}\right|_{nm}, \tag{A.0.2}$$

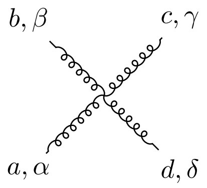
where the $\epsilon$ term (also known as the *Feynman parameter*) in the denominator preserves causality, $g^{\alpha\beta} = \text{diag}(1,-1,-1,-1)$ is the Minkowski matrix and the indices $m, n$ are those of the Dirac $\gamma$-matrices.

The 3 and 4-point gluon vertices are given by



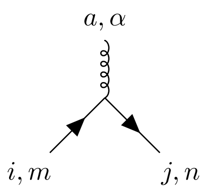
$$= g_s f^{abc}[g^{\alpha\beta}(p-q)^{\gamma} + g^{\beta\gamma}(q-r)^{\alpha} + g^{\gamma\alpha}(r-p)^{\beta}] \tag{A.0.3}$$

[2]All diagrams are drawn using TikZ-FeynHand [227] based on TikZ-Feynman [228].



$$\begin{aligned} &= -ig_s^2 f^{xac} f^{xbd} (g^{\alpha\beta} g^{\gamma\delta} - g^{\alpha\delta} g^{\beta\gamma}) \\ &\quad - ig_s^2 f^{xad} f^{xbc} (g^{\alpha\beta} g^{\gamma\delta} - g^{\alpha\gamma} g^{\beta\delta}) \\ &\quad - ig_s^2 f^{xab} f^{xcd} (g^{\alpha\gamma} g^{\beta\delta} - g^{\alpha\delta} g^{\beta\gamma}), \end{aligned} \tag{A.0.4}$$

and the $qqg$ vertex can be written as



$$= ig_s t^a_{ji} \gamma^\alpha_{nm}. \tag{A.0.5}$$

Finally, for diagrams containing loops, the momenta in the loop can take on any value without breaking momentum conservation between the initial and final states. We therefore include an integral for any loop present in the diagram. For example, for a loop with momentum $k$, the integral appearing in the matrix element would have the form

$$\int \frac{\mathrm{d}^d k}{(2\pi)^d}, \tag{A.0.6}$$

where we have not specified the the dimensionality of the integral at this stage to preserve generality. In addition, each fermion loop picks up a multiplier of $(-1)$.

The on-shell Ward identity governs the behaviour of the gluon polarisation vectors [229,230]. Unlike in QED, where this identity holds naturally because of the theory's Feynman rules, in QCD we must introduce the so-called *ghost fields* which are contained in the gauge fixing terms in Equation 2.1.1. As mentioned in Section 2.1, these additional fields are not considered in this thesis and so we omit the additional required Feynman Rules here. More details on this can be found in [28] and Chapter 16 in [24].

# Appendix B

# Monte Carlo Integration Algorithms

## B.1 RAMBO

The RAMBO algorithm [6] is the simplest implementation of a phase-space integrator and distributes points uniformly and isotropically. In the massless case, which we shall detail here, the algorithm generates 4-momenta with uniform weight by mapping the $[0,1]^{4n}$ hypercube of random numbers onto $n$ physical 4-momenta with c.o.m. energy $\sqrt{P^2}$.

To provide an implementation, we start by defining the phase-space volume for such a massless $n$ particle system

$$V_n = \int \prod_{i=1}^{n} \frac{\mathrm{d}^4 p_i}{(2\pi)^3} \theta(p_i^0) \delta(p_i^2) (2\pi)^4 \delta^{(4)} \left( P - \sum_{i=1}^{n} p_i \right), \tag{B.1.1}$$

where $P = (P, 0, 0, 0)$, $\theta$ is the Heaviside step function, the first Kronecker-$\delta$ ensures massless partons, and the latter momentum conservation. The RAMBO algorithm actually starts from a set of massless 4-momenta, $q_i^\mu$, which are not initially constrained by momentum conservation, but then become constrained through a series of operations. Specifically, we define the quantity

$$R_n = \int \prod_{i=1}^{n} \frac{\mathrm{d}q_i}{(2\pi)^3} \theta(q_i^0)\delta(q_i^2)(2\pi)^4 f(q_i^0) \tag{B.1.2}$$

$$= (2\pi)^{4-2n} \left( \int_0^\infty x f(x) \mathrm{d}x \right)^n, \tag{B.1.3}$$

where the function, $f$, ensures a finite volume phase-space. The $q_i^\mu$ are related to the physical $p_i^\mu$ through a Lorentz boost and scaling transformations

$$p_i^0 = x(\gamma q_i^0 + \vec{b} \cdot \vec{q}_i) \tag{B.1.4}$$

$$\vec{p}_i = x(\vec{q}_i + \vec{b}_i q_i^0 + a(\vec{b} \cdot \vec{q}_i)\vec{b}), \tag{B.1.5}$$

where

$$\vec{b} = -\frac{\vec{Q}}{M}, \;\; Q^\mu = \sum_{i=1}^{n} q_i^\mu, \;\; M = \sqrt{Q^2}, \tag{B.1.6}$$

$$x = \frac{\sqrt{P^2}}{M}, \;\; \gamma = \frac{Q^0}{M} = \sqrt{1 + \vec{b}}, \;\; a = \frac{1}{1+\gamma}. \tag{B.1.7}$$

In the above equations, and throughout this thesis, we only use the vector markings, $\vec{x}$, to denote the final three (spatial) components of the 4-momenta.

By choosing $f(x) = e^{-x}$, and using these transformations, we can arrive at the expression

$$R_n = V_n \cdot S_n, \tag{B.1.8}$$

with

$$S_n = 2\pi (P^2)^{2-n} \frac{\Gamma\left(\frac{3}{2}\right) \Gamma(n-1)\Gamma(2n)}{\Gamma\left(n + \frac{1}{2}\right)}, \tag{B.1.9}$$

where the Γ-function has the usual definition

$$\Gamma(n) = (n-1)!, \;\; \forall n \in \mathbb{Z}^+, \tag{B.1.10}$$

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \mathrm{d}x, \;\; \Re(z) > 0. \tag{B.1.11}$$

This naturally provides a Monte Carlo implementation for uniform phase-space point generation which consists of the following:

1. For a system of $n$ massless partons, generate $4n$ random numbers, $u_i$, which

are uniformly distributed in [0,1];

2. From these, generate $n$ massless 4-momenta, $q_i^\mu$, which are distributed isotropically and with energies, $q_i^0$, following the distribution $q_i^0 e^{-q_i^0} \mathrm{d}q_i^0$ by applying

$$c_i = 2u_{i_1} - 1, \;\; \psi_i = 2\pi u_{i_2}, \;\; q_i^0 = -\ln(u_{i3}u_{i4}), \tag{B.1.12}$$

$$q_i^x = q_i^0\sqrt{1-c_i^2}\,\cos\psi_i, \;\; q_i^y = q_i^0\sqrt{1-c_i^2}\,\sin\psi_i, \;\; q_i^z = q_i^0 c_i; \tag{B.1.13}$$

3. Transform the $q_i^\mu$ using the transformations given in Equations B.1.4 and B.1.5 to arrive at a set of physical 4-momenta $p_i^\mu$.

Since there is no dependence on the weight in Equation B.1.8, the event weights are constant. Therefore, this procedure provide a simple, flat phase-space sampling algorithm. This procedure can also be generalised to the massive case.

## B.2 VEGAS

The RAMBO algorithm described above gives a simple way to uniformly and isotropically distribute points in the 4-momentum phase-space of $n$ massless particles. While this algorithm is easily implemented, and is computationally inexpensive, this naive sampling of the phase-space may still lead to a large number of integrand evaluations having to be made to ensure convergence during integration. The VEGAS algorithm [73, 74] is commonly used in particle physics to optimise integration processes since it brings together many of the approaches from importance and stratified sampling (discussed in Section 3.3.2), while not requiring any knowledge of the integrand's behaviour a priori. It does this by sub-dividing the phase-space into smaller segments, sampling in each region, and then adjusts the sub-divisions accordingly to guide the sampler to place more points where the integrand is largest.

VEGAS normally begins by dividing the hypercube, $[0,1]^n$, into a regular grid which is then adapted to approximate the optimal probability distribution function (p.d.f.)

$$p_{\text{optimal}}(x) = \frac{|f(x)|}{\int_0^1 \mathrm{d}x |f(x)|} \tag{B.2.1}$$

by a step function in each segment. For computational memory reasons, VEGAS leverages the factorising structure of the p.d.f. [1]

$$p(x) = \prod_{i=1}^{n} p_i(x_i). \tag{B.2.2}$$

At each iteration, the estimator and variance can be calculated

$$\left\langle I_j \right\rangle = \frac{1}{N_j} \sum_{n=1}^{N_j} \frac{f(r_i)}{p(r_i)}, \;\; \sigma_j^2 = \frac{1}{N_j} \sum_{n=1}^{N_j} \left( \frac{f(r_i)}{p(r)} \right)^2 - \left\langle I_j \right\rangle^2 , \tag{B.2.3}$$

where $N_j$ is the number of sampling points in iteration $j$. Each subsequent iteration on a new grid layout is combined with the last so as to make efficient use of the integrand evaluations. This gives a global estimate of the integral

$$\langle I \rangle = \left( \sum_{j=1}^{m} \frac{N_j}{\sigma_j^2} \right)^{-1} \left( \sum_{j=1}^{m} \frac{N_j \left\langle I_j \right\rangle}{\sigma_j^2} \right) , \tag{B.2.4}$$

where $m$ is the total number of iterations and each estimate is weighted by the number of sampling points in that iteration, as well as the iteration's variance. A stopping criteria, such as the differences in inter-segment variance, is set to determine when an 'optimal grid' is found. This usually takes several iterations, but depends on the criteria and the complexity of the integrand.

During the optimisation steps, the variance can be unstable and so a damping term is introduced to help avoid this. In addition, the optimisation stage is computationally expensive as the adaptive stages require continuous variance calculations and grid adjustments, as well as phase-space evaluations. In practice, we use only a comparatively few points during the optimisation stage, "freeze" the optimised grid, and then use this for event generation with many points.

[1]While this limits the applicability of VEGAS to integrands which can factorise, this condition is commonly met in particle physics applications.

# Appendix C

# FKS pairs and partition functions

The FKS subtraction formalism was designed to provide a framework by which the divergent structure arising from the real radiation corrections at NLO can be constructed and subtracted in $(n{+}1)$ phase-space, where $n$ is the number of jets at the Born level, and added back in and solved analytically via dimensional regularisation [37]

$$\sigma_{\mathrm{NLO}} = \int \mathrm{d}\Phi_n \mathrm{d}\sigma^{(B)} + \int \mathrm{d}\Phi_n \left[\mathrm{d}\sigma^{(V)} + \int \mathrm{d}\Phi_1 \mathrm{d}\sigma^{(S)}\right] + \int \mathrm{d}\Phi_{n+1}[\mathrm{d}\sigma^{(R)} - \mathrm{d}\sigma^{(S)}], \tag{C.0.1}$$

where $\sigma^{(B)}$ is the Born cross-section, $\sigma^{(R)}$ and $\sigma^{(V)}$ are the real and virtual corrections at NLO and $\sigma^{(S)}$ is the real singular structure. By performing subtraction we are able to ensure that the singular structures of the virtual and real corrections cancel, thus leaving us with a non-divergent NLO cross-section.

For the processes considered here, the most general way of defining FKS pairs is given by

$$\begin{aligned}\mathcal{P}_{\mathrm{FKS}} = \{(i,j)\,|\, 1 \leq\ i \leq n_g + 2,\ 3 \leq\ j \leq n_g + 2, i \neq j, \\ \mathcal{M}^{(n+1,0)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i || \vec{p}_j\},\end{aligned} \tag{C.0.2}$$

which is the equivalent definition as that used in Equation (5.2.4), but where in Chapter 5, we used the pairs defined by the Born and virtual correction divergent structures, since we do not calculate real corrections and we are not trying to perform subtraction.

Given that FKS pairs are ordered, there is redundancy in Equation (C.0.2) since we will double count the soft singularities. An alternative definition is just to drop the $p_j^0 \to 0$ criteria to get

$$\mathcal{P}_{\text{FKS}} = \{(i,j) \,|\, 1 \leq\ i \leq n_g + 2,\ 3 \leq\ j \leq n_g + 2, i \neq j, \\ \mathcal{M}^{(n+1,0)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } \vec{p}_i || \vec{p}_j\}, \quad \text{(C.0.3)}$$

as shown in [38]. By using the definition given in Equation (C.0.3), we end up with the general FKS criteria that *each FKS partition contain at most one collinear and one soft singularity.* Formalising this mathematically allows us to require the following criteria be met by any such FKS partition function, $\mathcal{S}_{i,j}$ (adapted from [38])

$$\sum_{(i,j)\in\mathcal{P}_{\text{FKS}}} \mathcal{S}_{i,j} = 1, \quad \text{(C.0.4)}$$

$$\lim_{\vec{p}_k||\vec{p}_l} \mathcal{S}_{i,j} = 0,\ \forall(k,l) \in \mathcal{P}_{\text{FKS}} \text{ with } (k,l) \neq (i,j), \quad \text{(C.0.5)}$$

$$\lim_{p_k^0 \to 0} \mathcal{S}_{i,j} = 0,\ \forall(k,l) \in \mathcal{P}_{\text{FKS}} \text{ with } k \neq i. \quad \text{(C.0.6)}$$

Examples of partition functions satisfying these conditions are given in [38] in terms of energies and angles, and in [231] in terms of $s_{ij}$ variables among others.

While defining a function in terms of energies and angles can be beneficial when performing full FKS subtraction, for ease of computation we use the Lorentz invariant $s_{ij}$ variables defined in Equation (5.2.5). However, we note that the definition used in Equation (5.2.5) does not satisfy Equation (C.0.6) and therefore some of our partitions will contain multiple soft singularities and thus result in redundancies.

# Appendix D

# Hyperparameter tuning

This Appendix relates to hyperparameter tuning in the context of the gluon-induced diphoton amplitudes discussed in Chapter 6. Hyperparameter tuning was performed on a dataset of 1M points (derived independently from the datasets used for validation and testing in Section 6.4) to explore optimal data processing and model parameter choices. Given the computational expense of generating data, this was only done for the $2 \to 3$ process.

We tested different model architecture constructions (changing the number of hidden layers and/or the number of nodes in each hidden layer), data preprocessing methods, and model loss functions. All other training parameters are as described in Section 6.3.2. For data preprocessing methods, we tested input variable standardisation, i.e. the training and validation data input variables are each standardised to have zero mean and unit variance, and normalisation, i.e. the training and validation data input variables are each normalised according to min/max normalisation

$$x^* = \frac{x - \min(X_{\text{train}})}{\max(X_{\text{train}}) - \min(X_{\text{train}})} \tag{D.0.1}$$

where $x \in X_{\text{train}}$, $X_{\text{train}} \subset \mathbb{R}$ is the set of training data for a given input variable, and $x^*$ is the input variable normalised from $x$. This procedure means the dataset is normalised such that $x^* \in [0, 1]$ and therefore encourages a positive-definite output. When using the standardisation preprocessing step, we use hyperbolic-tangent activation functions in the hidden layers, but for normalisation we use rectified linear units (ReLU) [232]. This latter choice is to further encourage a positive-definite output and also aims to increase the rate of convergence.

It should be noted that a clear limitation of the positive-definite conditioning of the normalisation procedure is a reliance on the following conditions:

$$\min(X_{\text{train}}) = \min(X_{\text{train}} \cup X_{\text{test}}), \tag{D.0.2}$$

$$\max(X_{\text{train}}) = \max(X_{\text{train}} \cup X_{\text{test}}), \tag{D.0.3}$$

where $X_{\text{test}} \subset \mathbb{R}$ is the set variable inputs derived from the testing data, and therefore $X_{\text{train}} \cup X_{\text{test}}$ represents the combination of the training, validation and testing sets. Since the performance gain from using the ML approach is that the training and validation sets combined are much smaller than the testing set, the above conditions are likely to break down as $n(X_{\text{train}}) \ll n(X_{\text{test}})$.

Two model loss functions were tested during hyperparameter tuning. The first was the mean squared error (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{D.0.4}$$

where $n$ is the number of training points, $f : \mathbb{R}^d \to \mathbb{R}$ is the function describing the neural network, $x_i$ is the $i^{\text{th}}$ $d$-dimensional input data (here $d = 4n$), and $y_i$ the corresponding target variable. The second is the mean squared logarithmic error (MSLE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(\log(f(x_i) + 1) - \log(y_i + 1))^2. \tag{D.0.5}$$

Given the problem of approximating matrix element values for complex scattering processes, the target variable can take on a wide range of values spanning several orders of magnitude. After performing standardisation, there will still be important outliers in the tails of the standardised distribution of target variables. These large values can sometimes be especially important to the cross-section and so the MSE's penalisation of large outlier values can be beneficial; however, this might also make the training unstable. We included the MSLE during training to test if reducing the sensitivity to large scale variations in the target value is beneficial.

The results of the hyperparameter tuning can be found in Table D.1. Here we see that using data standardisation with an MSE loss function generally produces better results, although there does not seem to be a clear dependence on the model architecture or data processing method. Given these findings, we choose to train our models using data standardisation with hyperbolic-tangent activation functions, an MSE loss function, and an architecture of 20-40-20. This is consistent with the set up presented in Section 5.2.2.

| | Processing | | Layers | | | | Loss | | Error | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Std | Norm | 20-40-20 | 30-60-30 | 20-30-40-30-20 | 30-40-50-40-30 | MSE | MSLE | RMSE | RMSLE |
| 1 | x | | x | | | | x | | $2.828 \times 10^{-5}$ | $2.790 \times 10^{-5}$ |
| 2 | x | | x | | | | | x | $3.320 \times 10^{-5}$ | $3.288 \times 10^{-5}$ |
| 3 | x | | | x | | | x | | $2.829 \times 10^{-5}$ | $2.790 \times 10^{-5}$ |
| 4 | x | | | x | | | | x | $3.820 \times 10^{-5}$ | $3.791 \times 10^{-5}$ |
| 5 | x | | | | x | | x | | $2.829 \times 10^{-5}$ | $2.791 \times 10^{-5}$ |
| 6 | x | | | | x | | | x | $3.147 \times 10^{-5}$ | $3.113 \times 10^{-5}$ |
| 7 | x | | | | | x | x | | $2.830 \times 10^{-5}$ | $2.792 \times 10^{-5}$ |
| 8 | x | | | | | x | | x | $3.454 \times 10^{-5}$ | $3.422 \times 10^{-5}$ |
| 9 | | x | x | | | | x | | $2.835 \times 10^{-5}$ | $2.797 \times 10^{-5}$ |
| 10 | | x | x | | | | | x | $4.799 \times 10^{-4}$ | $4.802 \times 10^{-4}$ |
| 11 | | x | | x | | | x | | $2.835 \times 10^{-5}$ | $2.797 \times 10^{-5}$ |
| 12 | | x | | x | | | | x | $7.396 \times 10^{-4}$ | $7.405 \times 10^{-4}$ |
| 13 | | x | | | x | | x | | $2.836 \times 10^{-5}$ | $2.797 \times 10^{-5}$ |
| 14 | | x | | | x | | | x | $3.414 \times 10^{-4}$ | $3.416 \times 10^{-4}$ |
| 15 | | x | | | | x | x | | $2.836 \times 10^{-5}$ | $2.797 \times 10^{-5}$ |
| 16 | | x | | | | x | | x | $5.599 \times 10^{-4}$ | $5.604 \times 10^{-4}$ |

Table D.1: Hyperparameter tuning results. Tuning was performed on a fixed training dataset size of 100k points sampled using the RAMBO integrator [6] on a unit integration grid. Performance was measured with respect to both the Root Mean Squared Error (RMSE) and Root Mean Squared Logarithmic Error (RMSLE) so as to avoid biasing the error measure to the optimisation criterion (loss function) chosen.

# Appendix E

# $y_p$ tuning

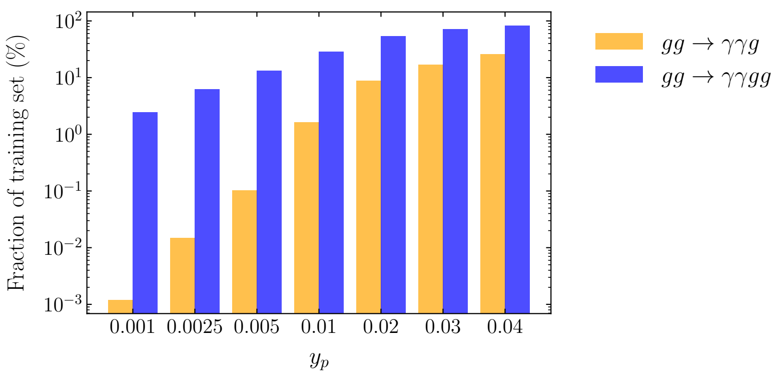


Figure E.1: Proportion of the training dataset in the divergent region, $\mathcal{R}_{\text{div}}$, as a function of $y_p$ for the $2 \to 3$ and $2 \to 4$ process.

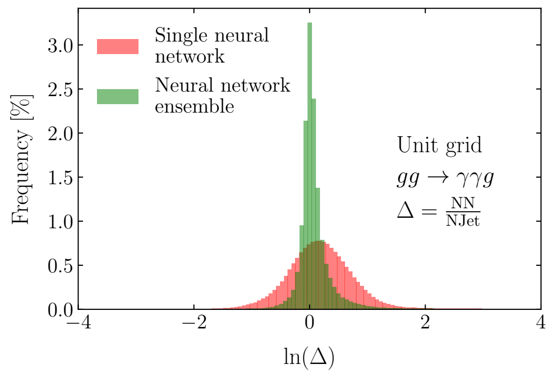
This Appendix relates to $y_p$ tuning in the context of the gluon-induced diphoton amplitudes discussed in Chapter 6. The choice of $y_p$ defines the partition between the divergent region of phase-space, $\mathcal{R}_{\text{div}}$, and the non-divergent region, $\mathcal{R}_{\text{non-div}}$ (see Equations 6.3.1 - 6.3.2). While it may be assumed that having more points in each region is helpful since it provides more data for the networks trained in each region, this is not always the case. Including a mixture of points in the training dataset, with large imbalances in the distribution of different scales, can make the network optimisation procedure increasingly noisy. For this reason, we seek to choose a value of $y_p$ which provides a balance between having enough divergent points to learn those regions well, whilst not providing too many points that are not in the limit and which share similar scales to points in the non-divergent regions of phase-space. We initially chose $y_p = 0.02$, although the number of points falling into $\mathcal{R}_{\text{div}}$ depends on the multiplicity of the process. As presented in Section 6.4.1, this value was shown to perform well, [1] yet the same value would place a significantly greater proportion of points into the divergent region when another external leg is added (see Figure E.1). Instead of choosing the same value of $y_p$ for all processes, we aim to select a value which keeps the proportion of points in the divergent region at the level of 2 - 8% of the whole phase-space sampled. We choose a value of $y_p = 0.02$ for the $2 \to 3$ process, and $y_p = 0.001$ for the $2 \to 4$ process. [2]

[1]The value of $y_p = 0.01$ was also tested and found to be in similarly good agreement.

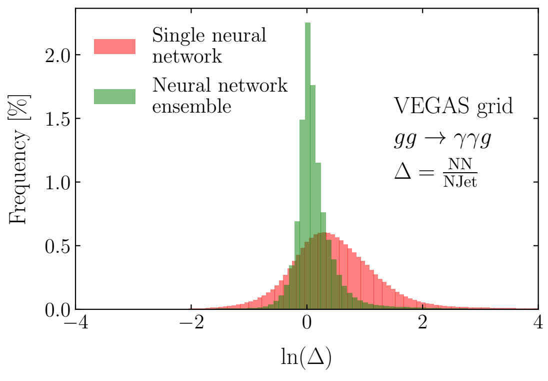
[2]A value of $y_p = 0.0025$ for the $2 \to 4$ process would also allow for this; however, at high multiplicity, the lower value of this cut provided more optimal performance.

# Appendix F

# Comparison with the naive setup



(a) Unit integration grid.



(b) VEGAS integration grid.

Figure F.1: Comparison of NN/NJet errors between the single NN and NN ensemble approaches for the $2 \to 3$ scattering process using different integration grids.

Throughout Chapter 6, all results that were presented using an ML approach have used the NN ensemble methodology. In Chapter 5, this approach was shown to outperform a naive single NN trained over the whole of phase-space for $e^+e^-$ collisions. In particular, the motivation for this approach was enhanced performance in handling real emission, IR singular regions of phase-space, which similarly occur in the processes studied in this work, especially at high multiplicity. For completeness, we perform a similar comparison on the $2 \to 3$ gluon-initiated diphoton processes; we do not compare on the $2 \to 4$ process as it is computationally expensive to do so and it is a natural higher multiplicity extension of the $2 \to 3$ process.

Figure F.1 shows the matrix level error analysis of the $2 \rightarrow 3$ scattering process, using both a unit and VEGAS optimisation grid. In both cases, the error distribution for the single NN approach has a significantly broader character than the ensemble method. This demonstrates that the findings described in Chapter 5 are consistent with those presented in this study.

# Bibliography

[1] P. D. Group, P. A. Zyla, R. M. Barnett, J. Beringer, O. Dahl, D. A. Dwyer et al., *Review of Particle Physics*, *Progress of Theoretical and Experimental Physics* **2020** (08, 2020) .

[2] 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].

[3] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning.* Prentice Hall, 2016.

[4] J.-F. Pekel, A. Cottam, N. Gorelick and A. S. Belward, *High-resolution mapping of global surface water and its long-term changes*, *Nature* **540** (2016) 418–422.

[5] "Covid-19 patient notification system (cpns) user guide." https://www.england.nhs.uk/statistics/wp-content/uploads/sites/2/2020/09/CPNS-User-Guide-20200831.pdf.

[6] 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.

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

[8] J. Aylett-Bullock, S. Badger and R. Moodie, *Optimising simulations for diphoton production at hadron colliders using amplitude neural networks*, *JHEP* **08** (2021) 66, [2106.09474].

[9] E. Nemni, J. Bullock, S. Belabbes and L. Bromley, *Fully convolutional neural network for rapid flood segmentation in synthetic aperture radar imagery*, *Remote Sensing* **12** (2020) .

[10] T. Logar, J. Bullock, E. Nemni, L. Bromley, J. A. Quinn and M. Luengo-Oroz, *Pulsesatellite: A tool using human-ai feedback loops for satellite image analysis in humanitarian contexts*, *Proceedings of the AAAI Conference on Artificial Intelligence* **34** (2020) 13628–13629.

[11] J. Bullock, A. Luccioni, K. Hoffman Pham, C. Sin Nga Lam and M. Luengo-Oroz, *Mapping the landscape of artificial intelligence applications against COVID-19*, *Journal of Artificial Intelligence Research* **69** (2020) 807–845, [2003.11336].

[12] M. Luengo-Oroz, K. Hoffmann Pham, J. Bullock, R. Kirkpatrick, A. Luccioni, S. Rubel et al., *Artificial intelligence cooperation to support the global response to COVID-19*, *Nature Machine Intelligence* **2** (2020) 295–297.

[13] A. Luccioni, J. Bullock, K. Hoffmann Pham, C. Sin Nga Lam and M. Luengo-Oroz, *Considerations, Good Practices, Risks and Pitfalls in Developing AI Solutions Against COVID-19*, in *Harvard CRCS Workshop on AI for Social Good*, 2020, 2008.09043.

[14] M. Luengo-Oroz, J. Bullock, K. Hoffmann Pham, C. Sin Nga Lam and A. Luccioni, *From artificial intelligence bias to inequality in the time of covid-19*, *IEEE Technology and Society Magazine* **40** (2021) 71–79.

[15] J. Aylett-Bullock, C. Cuesta-Lázaro, 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, [https://royalsocietypublishing.org/doi/pdf/10.1098/rsos.210506].

[16] J. Aylett-Bullock, C. Cuesta-Lázaro, A. Quera-Bofarull, A. Katta, K. Hoffmann Pham, B. Hoover et al., *Operational response simulation tool for epidemics within refugee and idp settlements*, *medRxiv* (2021) , [https://www.medrxiv.org/content/early/2021/01/29/2021.01.27.21250611.full

[17] J. Bullock, C. Cuesta-Lázaro and A. Quera-Bofarull, *XNet: a convolutional neural network (CNN) implementation for medical x-ray image segmentation suitable for small datasets*, in *Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging* (B. Gimi and A. Krol, eds.), vol. 10953, pp. 453 – 463, International Society for Optics and Photonics, SPIE, 2019, DOI.

[18] J. Bullock and M. Luengo-Oroz, *Automated Speech Generation from UN General Assembly Statements: Mapping Risks in AI Generated Texts*, in *International Conference on Machine Learning AI for Social Good Workshop*, ICML, 2019, 1906.01946.

[19] 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].

[20] 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].

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

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

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

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

[25] J. Campbell, J. Huston and F. Krauss, *The Black Book of Quantum Chromodynamics: A Primer for the LHC Era.* Oxford University Press, 12, 2017.

[26] R. K. Ellis, W. J. Stirling and B. R. Webber, *QCD and collider physics*, vol. 8. Cambridge University Press, 2, 2011.

[27] P. Skands, *Introduction to QCD*, in *Theoretical Advanced Study Institute in Elementary Particle Physics: Searching for New Physics at Small and Large Scales*, 7, 2012, `1207.2389`, DOI.

[28] M. L. Mangano, *Introduction to QCD*, in *1998 European School of High-Energy Physics*, 1998.

[29] T. Ohl, *Feynman Diagrams For Pedestrians*, in *46th Maria Laach School for High Energy Physics*, 2014, https://www.maria-laach.tp.nt.uni-siegen.de/downloads/files/2014/Ohl-2014.pdf.

[30] C. N. Yang and R. L. Mills, *Conservation of isotopic spin and isotopic gauge invariance*, *Phys. Rev.* **96** (Oct, 1954) 191–195.

[31] S. J. Parke and T. R. Taylor, *An Amplitude for $n$ Gluon Scattering*, *Phys. Rev. Lett.* **56** (1986) 2459.

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

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

[34] F. Bloch and A. Nordsieck, *Note on the radiation field of the electron*, *Phys. Rev.* **52** (Jul, 1937) 54–59.

[35] 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`].

[36] 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`].

[37] 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`].

[38] R. Frederix, S. Frixione, F. Maltoni and T. Stelzer, *Automation of next-to-leading order computations in QCD: The FKS subtraction*, *JHEP* **10** (2009) 003, [`0908.4272`].

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

[40] K. Symanzik, *Small distance behaviour in field theory and power counting*, *Communications in Mathematical Physics* **18** (1970) 227–246.

[41] A. Deur, S. J. Brodsky and G. F. de Teramond, *The QCD Running Coupling*, *Nucl. Phys.* **90** (2016) 1, [`1604.08082`].

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

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

[44] S. Weinzierl, *Introduction to Monte Carlo methods*, `hep-ph/0006269`.

[45] T. Sjostrand, S. Mrenna and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026, [hep-ph/0603175].

[46] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159–177, [1410.3012].

[47] M. Bahr et al., *Herwig++ Physics and Manual*, *Eur. Phys. J. C* **58** (2008) 639–707, [0803.0883].

[48] G. Corcella, I. G. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson et al., *HERWIG 6: An Event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)*, *JHEP* **01** (2001) 010, [hep-ph/0011363].

[49] J. Bellm et al., *Herwig 7.0/Herwig++ 3.0 release note*, *Eur. Phys. J. C* **76** (2016) 196, [1512.01178].

[50] Sherpa collaboration, E. Bothmann et al., *Event Generation with Sherpa 2.2*, *SciPost Phys.* **7** (2019) 034, [1905.09127].

[51] 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].

[52] F. Krauss, R. Kuhn and G. Soff, *AMEGIC++ 1.0: A Matrix element generator in C++*, *JHEP* **02** (2002) 044, [hep-ph/0109036].

[53] T. Gleisberg and S. Hoeche, *Comix, a new matrix element generator*, *JHEP* **12** (2008) 039, [0808.3674].

[54] T. Hahn and M. Perez-Victoria, *Automatized one loop calculations in four-dimensions and D-dimensions*, *Comput. Phys. Commun.* **118** (1999) 153–165, [hep-ph/9807565].

[55] A. van Hameren, *OneLOop: For the evaluation of one-loop scalar functions*, *Comput. Phys. Commun.* **182** (2011) 2427–2438, [1007.4716].

[56] R. K. Ellis and G. Zanderighi, *Scalar one-loop integrals for QCD*, *JHEP* **02** (2008) 002, [0712.1851].

[57] A. Denner, S. Dittmaier and L. Hofer, *COLLIER - A fortran-library for one-loop integrals*, *PoS* **LL2014** (2014) 071, [1407.0087].

[58] G. Ossola, C. G. Papadopoulos and R. Pittau, *CutTools: A Program implementing the OPP reduction method to compute one-loop amplitudes*, *JHEP* **03** (2008) 042, [0711.3596].

[59] T. Peraro, *Ninja: Automated Integrand Reduction via Laurent Expansion for One-Loop Amplitudes*, *Comput. Phys. Commun.* **185** (2014) 2771–2797, [1403.1229].

[60] P. Mastrolia, G. Ossola, T. Reiter and F. Tramontano, *Scattering AMplitudes from Unitarity-based Reduction Algorithm at the Integrand-level*, *JHEP* **08** (2010) 080, [1006.0710].

[61] 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].

[62] 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].

[63] J. M. Campbell, R. K. Ellis and C. Williams, *Vector boson pair production at the LHC*, *JHEP* **07** (2011) 018, [1105.0020].

[64] J. M. Campbell, R. K. Ellis and W. T. Giele, *A Multi-Threaded Version of MCFM*, *Eur. Phys. J. C* **75** (2015) 246, [1503.06182].

[65] 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].

[66] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, G. Ossola et al., *GoSam: A Program for Automated One-Loop Calculations*, *J. Phys. Conf. Ser.* **368** (2012) 012056, [1111.6534].

[67] 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].

[68] 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].

[69] F. Cascioli, P. Maierhofer and S. Pozzorini, *Scattering Amplitudes with Open Loops*, *Phys. Rev. Lett.* **108** (2012) 111601, [1111.5206].

[70] 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].

[71] Z. Nagy and D. E. Soper, *Matching parton showers to NLO computations*, *JHEP* **10** (2005) 024, [hep-ph/0503053].

[72] S. Hoeche, F. Krauss, N. Lavesson, L. Lonnblad, M. Mangano, A. Schalicke et al., *Matching parton showers and matrix elements*, in *HERA and the LHC: A Workshop on the Implications of HERA for LHC Physics: CERN - DESY Workshop 2004/2005 (Midterm Meeting, CERN, 11-13 October 2004; Final Meeting, DESY, 17-21 January 2005)*, 2005, hep-ph/0602031, DOI.

[73] G. P. Lepage, *VEGAS: An Adaptive Multidimensional Integration Program*, 1980.

[74] T. Ohl, *Vegas revisited: Adaptive Monte Carlo integration beyond factorization*, *Comput. Phys. Commun.* **120** (1999) 13–19, [`hep-ph/9806432`].

[75] R. Kleiss and R. Pittau, *Weight optimization in multichannel Monte Carlo*, *Comput. Phys. Commun.* **83** (1994) 141–146, [`hep-ph/9405257`].

[76] A. van Hameren and C. G. Papadopoulos, *A Hierarchical phase space generator for QCD antenna structures*, *Eur. Phys. J.* **C25** (2002) 563–574, [`hep-ph/0204055`].

[77] P. D. Draggiotis, A. van Hameren and R. Kleiss, *SARGE: An Algorithm for generating QCD antennas*, *Phys. Lett.* **B483** (2000) 124–130, [`hep-ph/0004047`].

[78] S. Forte, L. Garrido, J. I. Latorre and A. Piccione, *Neural network parametrization of deep inelastic structure functions*, *JHEP* **05** (2002) 062, [`hep-ph/0204232`].

[79] M. Feickert and B. Nachman, *A Living Review of Machine Learning for Particle Physics*, `2102.02770`.

[80] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* MIT Press, 2010.

[81] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification And Regression Trees.* Routledge, 1984.

[82] S. K. Murthy, *Automatic construction of decision trees from data: A multi-disciplinary survey*, *Data Mining and Knowledge Discovery* **2** (1998) 345–389.

[83] L. Breiman, *Random forests*, *Machine Learning* **45** (2001) 5–32.

[84] J. H. Friedman, *Greedy function approximation: A gradient boosting machine*, *The Annals of Statistics* **29** (2001) 1189–1232.

[85] J. H. Friedman, *Stochastic gradient boosting*, *Comput. Stat. Data Anal.* **38** (2002) 367–378.

[86] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, `1707.00028`.

[87] 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.

[88] F. Bishara and M. Montull, *(Machine) Learning amplitudes for faster event generation*, `1912.11055`.

[89] M. Minsky and S. Papert, *Perceptrons.* M.I.T. Press, 1969.

[90] S. Ruder, *An overview of gradient descent optimization algorithms*, *CoRR* **abs/1609.04747** (2016) , [`1609.04747`].

[91] J. Duchi, E. Hazan and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, *Journal of Machine Learning Research* **12** (2011) 2121–2159.

[92] N. N. for Machine Learning, *Lecture 6e RMSProp: Divide the gradient by a running average of its recent magnitude*, *Coursera: Neural networks for machine learning* (2012) .

[93] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *3rd International Conference for Learning Representations* (2015) , [`1412.6980`].

[94] B. Polyak, *Some methods of speeding up the convergence of iteration methods*, *USSR Computational Mathematics and Mathematical Physics* **4** (1964) 1–17.

[95] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, *Activation functions: Comparison of trends in practice and research for deep learning*, *CoRR* **abs/1811.03378** (2018) , [`1811.03378`].

[96] Y. A. LeCun, L. Bottou, G. B. Orr and K.-R. Müller, *Efficient BackProp*, pp. 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[97] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind, *Automatic differentiation in machine learning: A survey*, *J. Mach. Learn. Res.* **18** (2017) 5595–5637.

[98] M. D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, *SciPost Phys.* **9** (2020) 053, [`1810.11509`].

[99] I.-K. Chen, M. D. Klimek and M. Perelstein, *Improved Neural Network Monte Carlo Simulation*, *SciPost Phys.* **10** (2021) 023, [`2009.07819`].

[100] 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`].

[101] 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`].

[102] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event Generation with Normalizing Flows*, *Phys. Rev. D* **101** (2020) 076002, [`2001.10028`].

[103] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1530–1538, JMLR.org, 2015.

[104] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear independent components estimation*, in *3rd International Conference on Learning Representations*, ICLR, 2015.

[105] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using real nvp*, in *5th International Conference on Learning Representations*, ICLR, 2017.

[106] B. Uria, M.-A. Côté, K. Gregor, I. Murray and H. Larochelle, *Neural autoregressive distribution estimation*, *Journal of Machine Learning Research* **17** (2016) 1–37.

[107] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling, *Improved variational inference with inverse autoregressive flow*, in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 4743–4751, Curran Associates Inc., 2016.

[108] G. Papamakarios, T. Pavlakou and I. Murray, *Masked autoregressive flow for density estimation*, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 2335–2344, Curran Associates Inc., 2017.

[109] B. Stienen and R. Verheyen, *Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows*, *SciPost Phys.* **10** (2021) 038, [2011.13445].

[110] E. Bothmann and L. Debbio, *Reweighting a parton shower using a neural network: the final-state case*, *JHEP* **01** (2019) 033, [1808.07802].

[111] J. W. Monk, *Deep Learning as a Parton Shower*, *JHEP* **12** (2018) 021, [1807.03685].

[112] K. Dohi, *Variational Autoencoders for Jet Simulation*, 2009.04842.

[113] B. Nachman and J. Thaler, *Neural resampler for Monte Carlo reweighting with preserved uncertainties*, *Phys. Rev. D* **102** (2020) 076004, [2007.11586].

[114] 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.* **C80** (2020) 12, [1810.08312].

[115] 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].

[116] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair et al., *Generative adversarial nets*, in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger, eds.), pp. 2672–2680. Curran Associates, Inc., 2014.

[117] I. J. Goodfellow, *On distinguishability criteria for estimating generative models*, in *3rd International Conference on Learning Representations*, ICLR, 2015.

[118] L. Mescheder, A. Geiger and S. Nowozin, *Which training methods for GANs do actually converge?*, in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3481–3490, PMLR, 10–15 Jul, 2018, http://proceedings.mlr.press/v80/mescheder18a.html.

[119] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen et al., *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*, 1901.00875.

[120] SHiP collaboration, C. Ahdida et al., *Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks*, *JINST* **14** (2019) P11028, [1909.04451].

[121] J. Lin, W. Bhimji and B. Nachman, *Machine Learning Templates for QCD Factorization in the Search for Physics Beyond the Standard Model*, *JHEP* **05** (2019) 181, [1903.02556].

[122] 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].

[123] T. Lebese, B. Mellado and X. Ruan, *The use of Generative Adversarial Networks to characterise new physics in multi-lepton final states at the LHC*, 2105.14933.

[124] 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** (2020) 110, [1903.02433].

[125] Y. LeCun, *Generalization and network design strategies*, *Connectionism in Perspective* (1989) .

[126] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard et al., *Backpropagation applied to handwritten zip code recognition*, *Neural Computation* **1** (1989) .

[127] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, *LHC analysis-specific datasets with Generative Adversarial Networks*, 1901.05282.

[128] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC Events*, *SciPost Phys.* **7** (2019) 075, [1907.03764].

[129] Y. Alanazi et al., *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, 2001.11103.

[130] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying Event Samples*, 2008.06545.

[131] M. Backes, A. Butter, T. Plehn and R. Winterhalder, *How to GAN Event Unweighting*, *SciPost Phys.* **10** (2021) 089, [2012.07873].

[132] A. Butter, T. Plehn and R. Winterhalder, *How to GAN Event Subtraction*, 1912.08824.

[133] M. Bellagente, M. Haußmann, M. Luchmann and T. Plehn, *Understanding Event-Generation Networks via Uncertainties*, 2104.04543.

[134] G. Kasieczka, M. Luchmann, F. Otterpohl and T. Plehn, *Per-Object Systematics using Deep-Learned Calibration*, *SciPost Phys.* **9** (2020) 089, [2003.11099].

[135] A. Butter and T. Plehn, *Generative Networks for LHC events*, 2008.08558.

[136] A. Butter and T. Plehn, *Generative Models in Event Simulation*, *PoS* **LHCP2020** (2021) 055.

[137] HEP Software Foundation collaboration, J. Apostolakis et al., *HEP Software Foundation Community White Paper Working Group - Detector Simulation*, 1803.04165.

[138] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2014.

[139] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, *Foundations and Trends® in Machine Learning* **12** (2019) 307–392.

[140] 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].

[141] 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].

[142] M. Czakon, *Tops from Light Quarks: Full Mass Dependence at Two-Loops in QCD*, *Phys. Lett.* **B664** (2008) 307–314, [0803.1400].

[143] S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk et al., *Higgs Boson Pair Production in Gluon Fusion at Next-to-Leading Order with Full Top-Quark Mass Dependence*, *Phys. Rev. Lett.* **117** (2016) 012001, [1604.06447].

[144] G. Heinrich, S. P. Jones, M. Kerner, G. Luisoni and E. Vryonidou, *NLO predictions for Higgs boson pair production with full top quark mass dependence matched to parton showers*, *JHEP* **08** (2017) 088, [1703.09252].

[145] S. P. Jones, M. Kerner and G. Luisoni, *Next-to-Leading-Order QCD Corrections to Higgs Boson Plus Jet Production with Full Top-Quark Mass Dependence*, *Phys. Rev. Lett.* **120** (2018) 162001, [1802.00349].

[146] G. Heinrich, S. P. Jones, M. Kerner, G. Luisoni and L. Scyboz, *Probing the trilinear Higgs boson coupling in di-Higgs production at NLO QCD including parton shower effects*, *JHEP* **06** (2019) 066, [1903.08137].

[147] G. Ossola, C. G. Papadopoulos and R. Pittau, *Reducing full one-loop amplitudes to scalar integrals at the integrand level*, *Nucl. Phys.* **B763** (2007) 147–169, [hep-ph/0609007].

[148] Z. Bern, L. J. Dixon, D. C. Dunbar and D. A. Kosower, *Fusing gauge theory tree amplitudes into loop amplitudes*, *Nucl. Phys.* **B435** (1995) 59–101, [hep-ph/9409265].

[149] R. Britto, F. Cachazo and B. Feng, *Generalized unitarity and one-loop amplitudes in N=4 super-Yang-Mills*, *Nucl. Phys.* **B725** (2005) 275–305, [hep-th/0412103].

[150] R. K. Ellis, W. T. Giele and Z. Kunszt, *A Numerical Unitarity Formalism for Evaluating One-Loop Amplitudes*, *JHEP* **03** (2008) 003, [0708.2398].

[151] W. T. Giele, Z. Kunszt and K. Melnikov, *Full one-loop amplitudes from tree amplitudes*, *JHEP* **04** (2008) 049, [0801.2237].

[152] D. Forde, *Direct extraction of one-loop integral coefficients*, *Phys. Rev.* **D75** (2007) 125019, [0704.1835].

[153] S. D. Badger, *Direct Extraction Of One Loop Rational Terms*, *JHEP* **01** (2009) 049, [0806.4600].

[154] F. A. Berends and W. T. Giele, *Recursive Calculations for Processes with n Gluons*, *Nucl. Phys.* **B306** (1988) 759–808.

[155] 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].

[156] 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].

[157] M. Czakon and D. Heymes, *Four-dimensional formulation of the sector-improved residue subtraction scheme*, *Nucl. Phys.* **B890** (2014) 152–227, [1408.2500].

[158] JADE collaboration, W. Bartel et al., *Experimental Studies on Multi-Jet Production in e+ e- Annihilation at PETRA Energies*, *Z. Phys.* **C33** (1986) 23.

[159] R. Frederix, S. Frixione, K. Melnikov and G. Zanderighi, *NLO QCD corrections to five-jet production at LEP and the extraction of $\alpha_s(M_Z)$*, *JHEP* **11** (2010) 050, [1008.5313].

[160] F. Chollet et al., “Keras.” `https://github.com/fchollet/keras`, 2015.

[161] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., “TensorFlow: Large-scale machine learning on heterogeneous systems.” `https://www.tensorflow.org/`, 2015.

[162] N. Tagasovska and D. Lopez-Paz, *Single-model uncertainties for deep learning*, *NeurlPS* (2019) , [1811.00908].

[163] Y. Gal, *Uncertainty in Deep Learning*, Ph.D. thesis, University of Cambridge, 2016.

[164] B. Nachman, *A guide for deploying Deep Learning in LHC searches: How to achieve optimality and account for uncertainty*, 1909.03081.

[165] B. Nachman and C. Shimmin, *AI Safety for High Energy Physics*, 1910.08606.

[166] S. Bollweg, M. Haußmann, G. Kasieczka, M. Luchmann, T. Plehn and J. Thompson, *Deep-Learning Jets with Uncertainties and More*, 1904.10004.

[167] C. Englert, P. Galler, P. Harris and M. Spannowsky, *Machine Learning Uncertainties with Adversarial Neural Networks*, *Eur. Phys. J.* **C79** (2019) 4, [1807.08763].

[168] T. Gehrmann, N. Greiner and G. Heinrich, *Precise QCD predictions for the production of a photon pair in association with two jets*, *Phys. Rev. Lett.* **111** (2013) 222002, [1308.3660].

[169] S. Badger, A. Guffanti and V. Yundin, *Next-to-leading order QCD corrections to di-photon production in association with up to three jets at the Large Hadron Collider*, *JHEP* **03** (2014) 122, [1312.5927].

[170] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Hoeche, H. Ita, D. A. Kosower et al., *Next-to-leading order $\gamma\gamma + 2$-jet production at the LHC*, *Phys. Rev. D* **90** (2014) 054004, [1402.4127].

[171] B. Agarwal, F. Buccioni, A. von Manteuffel and L. Tancredi, *Two-loop leading colour QCD corrections to $q\bar{q} \to \gamma\gamma g$ and $qg \to \gamma\gamma q$*, *JHEP* **04** (2021) 201, [2102.01820].

[172] H. A. Chawdhry, M. Czakon, A. Mitov and R. Poncelet, *Two-loop leading-colour QCD helicity amplitudes for two-photon plus jet production at the LHC*, 2103.04319.

[173] B. Agarwal, F. Buccioni, A. von Manteuffel and L. Tancredi, *Two-loop helicity amplitudes for diphoton plus jet production in full color*, 2105.04585.

[174] H. A. Chawdhry, M. Czakon, A. Mitov and R. Poncelet, *NNLO QCD corrections to diphoton production with an additional jet at the LHC*, 2105.06940.

[175] S. Badger, C. Brønnum-Hansen, D. Chicherin, T. Gehrmann, H. B. Hartanto, J. Henn et al., *Virtual QCD corrections to gluon-initiated diphoton plus jet production at hadron colliders*, 2106.08664.

[176] D. de Florian and Z. Kunszt, *Two photons plus jet at LHC: The NNLO contribution from the g g initiated process*, *Phys. Lett. B* **460** (1999) 184–188, [hep-ph/9905283].

[177] Z. Bern, A. De Freitas and L. J. Dixon, *Two loop amplitudes for gluon fusion into two photons*, *JHEP* **09** (2001) 037, [hep-ph/0109078].

[178] T. Peraro, *FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs*, *JHEP* **07** (2019) 031, [1905.08019].

[179] Z. Bern, L. J. Dixon and D. A. Kosower, *One loop corrections to five gluon amplitudes*, *Phys. Rev. Lett.* **70** (1993) 2677–2680, [hep-ph/9302280].

[180] E. Byckling and K. Kajantie, *Particle Kinematics: (Chapters I-VI, X).* University of Jyvaskyla, Jyvaskyla, Finland, 1971.

[181] M. Cacciari, G. P. Salam and G. Soyez, *The anti-$k_t$ jet clustering algorithm*, *JHEP* **04** (2008) 063, [0802.1189].

[182] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896, [1111.6097].

[183] S. Frixione, *Isolated photons in perturbative QCD*, *Phys. Lett. B* **429** (1998) 369–374, [hep-ph/9801442].

[184] G. Guennebaud, B. Jacob et al., “Eigen v3.” http://eigen.tuxfamily.org, 2010.

[185] A. Buckley, J. Butterworth, D. Grellscheid, H. Hoeth, L. Lonnblad, J. Monk et al., *Rivet user manual*, *Comput. Phys. Commun.* **184** (2013) 2803–2819, [1003.0694].

[186] C. Bierlich et al., *Robust Independent Validation of Experiment and Theory: Rivet version 3*, *SciPost Phys.* **8** (2020) 026, [1912.05451].

[187] ATLAS collaboration, M. Aaboud et al., *Measurements of integrated and differential cross sections for isolated photon pair production in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector*, *Phys. Rev. D* **95** (2017) 112005, [1704.03839].

[188] 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].

[189] NNPDF collaboration, R. D. Ball et al., *Parton distributions from high-precision collider data*, *Eur. Phys. J. C* **77** (2017) 663, [1706.00428].

[190] S. Lang, P. Füreder, O. Kranz, B. Card, S. Roberts and A. Papp, *Humanitarian emergencies: causes, traits and impacts as observed by remote sensing*, in *Remote Sensing of Water Resources, Disasters, and Urban Studies*, pp. 483–512. CRC Press, 2015.

[191] J. A. Quinn, M. M. Nyhan, C. Navarro, D. Coluccia, L. Bromley and M. Luengo-Oroz, *Humanitarian applications of machine learning with remote-sensing data: review and case study in refugee settlement mapping*, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **376** (2018) 20170363, [https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2017.0363].

[192] K. He, G. Gkioxari, P. Dollár and R. B. Girshick, *Mask r-cnn*, *2017 IEEE International Conference on Computer Vision (ICCV)* (2017) 2980–2988.

[193] "Centre for research on the epidemiology of disasters. the human cost of weather-related disasters 1995-2015." United Nations Office for Disaster Risk Reduction, 2015.

[194] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek and K. P. Papathanassiou, *A tutorial on synthetic aperture radar*, *IEEE Geoscience and Remote Sensing Magazine* **1** (2013) 6–43.

[195] F.Covello, F. Battazza, A. Coletta, E. Lopinto, C. Fiorentino, L. Pietranera et al., *Cosmo-skymed an existing opportunity for observing the earth*, *Journal of Geodynamics* **49** (2010) 171 – 180.

[196] R. Werninghaus, *TerraSAR-X mission*, in *SAR Image Analysis, Modeling, and Techniques VI* (F. Posa, ed.), vol. 5236, pp. 9 – 16, International Society for Optics and Photonics, SPIE, 2004, DOI.

[197] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, *CoRR* **abs/1505.04597** (2015) .

[198] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[199] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[200] R. E. Russell, R. A. Katz, K. Richgels, D. P. Walsh and E. Grant, *A framework for modeling emerging diseases to inform management*, *Emerging Infectious Diseases* **23** (2017) 1–6.

[201] N. Brandon, K. L. Dionisio, K. Isaacs, R. Tornero-Velez, D. Kapraun, R. W. Setzer et al., *Simulating exposure-related behaviors using agent-based models*

*embedded with needs-based artificial intelligence*, *Journal of exposure science & environmental epidemiology* (2018) 1–10.

[202] A. H. Auchincloss, S. Y. Gebreab, C. Mair and A. V. Diez Roux, *A review of spatial methods in epidemiology, 2000–2010*, *Annual review of public health* **33** (2012) 107–122.

[203] A. M. El-Sayed, P. Scarborough, L. Seemann and S. Galea, *Social network analysis and agent-based modeling in social epidemiology*, *Epidemiologic Perspectives & Innovations* **9** (2012) 1.

[204] R. J. Rockett, A. Arnott, C. Lam, R. Sadsad, V. Timms, K.-A. Gray et al., *Revealing covid-19 transmission in australia by sars-cov-2 genome sequencing and agent-based modeling*, *Nature medicine* **26** (2020) 1398–1404.

[205] N. M. Ferguson, D. A. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley and D. S. Burke, *Strategies for mitigating an influenza pandemic*, *Nature* **442** (2006) 448–452.

[206] D. L. Chao, M. E. Halloran, V. J. Obenchain and I. M. Longini Jr, *Flute, a publicly available stochastic influenza epidemic simulation model*, *PLoS Comput Biol* **6** (2010) e1000656.

[207] E. J. Williamson, A. J. Walker, K. Bhaskaran, S. Bacon, C. Bates, C. E. Morton et al., *Factors associated with covid-19-related death using opensafely*, *Nature* **584** (2020) 430–436.

[208] J. Mossong, N. Hens, M. Jit, P. Beutels, K. Auranen, R. Mikolajczyk et al., *Social contacts and mixing patterns relevant to the spread of infectious diseases*, *PLoS Med* **5** (2008) e74.

[209] P. Klepac, S. Kissler and J. Gog, *Contagion! the bbc four pandemic – the model behind the documentary*, *Epidemics* **24** (2018) 49 – 59.

[210] P. S. Craig, M. Goldstein, A. H. Seheult and J. A. Smith, *Pressure matching for hydrocarbon reservoirs: a case study in the use of bayes linear strategies for large computer experiments (with discussion)*, in *Case Studies in Bayesian Statistics* (C. Gatsonis, J. S. Hodges, R. E. Kass, R. McCulloch, P. Rossi and N. D. Singpurwalla, eds.), vol. 3, pp. 36–93. SV, New York, 1997.

[211] I. Vernon, M. Goldstein and R. G. Bower, *Galaxy formation: a bayesian uncertainty analysis*, *Bayesian Analysis* **5** (2010) 619–670.

[212] I. Andrianakis, I. Vernon, N. McCreesh, T. McKinley, J. Oakley, R. Nsubuga et al., *Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on HIV in uganda.*, *PLoS Comput Biol.* **11** (2015) e1003968.

[213] K. Shea, R. K. Borchering, W. J. M. Probert, E. Howerton, T. L. Bogich, S. Li et al., *COVID-19 reopening strategies at the county level in the face of uncertainty: Multiple models for outbreak decision support*, *medRxiv* (2020) , [`https://www.medrxiv.org/content/early/2020/11/05/2020.11.03.20225409.full`

[214] D. K. Chu, E. A. Akl, S. Duda, K. Solo, S. Yaacoub and H. J. Schünemann, *Physical distancing, face masks, and eye protection to prevent person-to-person transmission of sars-cov-2 and covid-19: a systematic review and meta-analysis*, *Lancet* (2020) 1973–1987.

[215] J. Howard, A. Huang, Z. Li, Z. Tufekci, V. Zdimal, H.-M. van der Westhuizen et al., *Face masks against COVID-19: an evidence review*, .

[216] E. P. Fischer, M. C. Fischer, D. Grass, I. Henrion, W. S. Warren and E. Westman, *Low-cost measurement of face mask efficacy for filtering expelled droplets during speech*, *Science Advances* **6** (2020) eabd3083.

[217] Y. Wang, H. Tian, L. Zhang, M. Zhang, D. Guo, W. Wu et al., *Reduction of secondary transmission of SARS-CoV-2 in households by face mask use,*

*disinfection and social distancing: a cohort study in Beijing, China*, *BMJ Global Health* **5** (2020) e002794.

[218] J. Atkinson, Y. Chartier, C. L. Pessoa-Silva, P. Jensen, Y. Li and W.-H. Seto, *Natural ventilation for infection control in health-care settings.* World Health Organization, 2009.

[219] S. Carrazza, J. Cruz-Martinez, M. Rossi and M. Zaro, *MadFlow: automating Monte Carlo simulation on GPU for particle physics processes*, 2106.10279.

[220] S. Carrazza and J. M. Cruz-Martinez, *VegasFlow: accelerating Monte Carlo simulation across multiple hardware platforms*, *Comput. Phys. Commun.* **254** (2020) 107376, [2002.12921].

[221] J. M. Cruz-Martinez and S. Carrazza, *VegasFlow: accelerating Monte Carlo simulation across platforms*, *PoS* **ICHEP2020** (2021) 906, [2010.09341].

[222] S. Carrazza, J. M. Cruz-Martinez and M. Rossi, *PDFFlow: Parton distribution functions on GPU*, *Comput. Phys. Commun.* **264** (2021) 107995, [2009.06635].

[223] M. Rossi, S. Carrazza and J. M. Cruz-Martinez, *PDFFlow: hardware accelerating parton density access*, *PoS* **ICHEP2020** (2021) 921, [2012.08221].

[224] S. Carrazza, J. Cruz-Martinez, M. Rossi and M. Zaro, *Towards the automation of Monte Carlo simulation on GPU for particle physics processes*, in *25th International Conference on Computing in High-Energy and Nuclear Physics*, 5, 2021, 2105.10529.

[225] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower et al., *Ntuples for NLO Events at Hadron Colliders*, *Comput. Phys. Commun.* **185** (2014) 1443–1460, [1310.7439].

[226] D. Maitre, G. Heinrich and M. Johnson, *N(N)LO event files: applications and prospects*, *PoS* **LL2016** (2016) 016, [1607.06259].

[227] M. Dohse, *TikZ-FeynHand: Basic User Guide*, 1802.00689.

[228] J. Ellis, *TikZ-Feynman: Feynman diagrams with TikZ*, *Comput. Phys. Commun.* **210** (2017) 103–123, [1601.05437].

[229] J. C. Ward, *An Identity in Quantum Electrodynamics*, *Phys. Rev.* **78** (1950) 182.

[230] Y. Takahashi, *On the generalized Ward identity*, *Nuovo Cim.* **6** (1957) 371.

[231] S. Frixione, E. Laenen, P. Motylinski and B. R. Webber, *Single-top production in MC@NLO*, *JHEP* **03** (2006) 092, [hep-ph/0512250].

[232] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 807–814, Omnipress, 2010.